Flash i XML.
Techniki zaawansowane

Tytuł oryginału: Flash XML StudioLab
Autorzy: Ian Tindale, Paul Macdonald, James Rowley
Tłumaczenie: Marek Binkowski
ISBN: 83-7197-657-7
Format: B5 Stron: 452

Ta książka ukazuje nietypowe spojrzenie na specyficzną cechę Flasha 5 – możliwość współpracy z językiem XML. Do zrozumienia jej treści przydatne jest opanowanie podstaw pracy z Flashem 5 i językiem programowania ActionScript.

Współpraca Flasha 5 z językiem XML daje ogromne możliwości, na których skoncentrujemy się w tej książce. Nie zakładamy, że miałeś wcześniej jakikolwiek kontakt z językiem XML. W pierwszych rozdziałach omówino elementarne zagadnienia związane z tym językiem. Integralną częścią książki jest praktyczne studium, którego części prezentujemy w kolejnych rozdziałach. Śledząc ich treść, szybko zdobędziesz umiejętności potrzebne do wykorzystania wszechstronnych zastosowań języka XML we Flashu. W dalszych rozdziałach zawartość studium odejdzie nieco od treści rozdziałów, lecz nie przejmuj się – przez cały czas będziesz zdobywał nową wiedzę i umiejętności, dzięki którym będziesz mógł tworzyć interesujące aplikacje, korzystające z technologii Flash XML.

W ostatnich trzech rozdziałach zaprezentujemy prawdziwie dynamiczne zastosowania, możliwe wówczas, gdy aplikacje porzucają ograniczenia pojedynczego komputera i zaczynają działać w sieci. Korzystaj bez ograniczeń z prezentowanych przykładów i adaptuj je do własnych potrzeb. Poznaj pasjonujące możliwości współpracy języka XML i Flasha.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 12.
Flash, XML i bazy danych

Gdy tworzysz witryny internetowe we Flashu, prędzej czy później będziesz zmuszony do komunikacji z bazami danych. Obawiam się, że przed tym nie ma ucieczki. Jeśli jesteś podobny do mnie, pewnie myślisz, że programowanie baz danych jest niesamowicie nudnym procesem, sprawiającym radość jedynie zarośniętym guru, bawiącym się gigantycznymi serwerami. Przyszedł jednak moment w mojej karierze, gdy szybko zmieniłem zdanie. Być może nadszedł czas, byś i ty je zmienił.

Statyczne dokumenty, czy to w formacie HTML czy XML, po prostu nie nadają się do niektórych zastosowań sieciowych, wymagających wyświetlania dynamicznych, ciągle aktualizowanych materiałów. Niektórzy analitycy twierdzą, że Internet stopniowo staje się interfejsem dla materiałów, które zwykle są przechowywane w bazach danych.

Wiele witryn internetowych, które utworzysz w przyszłości, będzie wymagało zapamiętania informacji wpisywanych przez użytkowników w bazie danych. Oczywiście później dane zapisane w bazie trzeba będzie w jakiś sposób wyświetlić — czyli popłyną w drugą stronę, do dynamicznej witryny. Pracując we Flashu, będziesz mógł przygotować dla danych niemal dowolny interfejs, jaki ci przyjdzie do głowy.

Połączenie witryn internetowych z bazami danych można interpretować z różnej perspektywy:

- Użytkowania baz danych do dostarczania materiałów witrynie internetowej.
- Użytkowania witryn w celu udostępnienia użytkownikom bazy danych.

Pierwsze podejście, czyli korzystanie z baz danych, umożliwia tworzenie witryn, które oferują użytkownikowi bezproblemowy dostęp do danych zawartych w bazie i których utrzymanie i aktualizacja nie jest trudna. Drugie podejście, czyli użytkowanie witryn jako interfejsów bazy danych umożliwia projektantowi tworzenie przyjaznych, międzyplatformowych frontonów udostępniających dane użytkownikom w dowolnym miejscu na świecie.

W tym rozdziale pokażemy, jak przygotować wszystkie elementy niezbędne do uruchomienia aplikacji flashowej, wyświetlającej materiały pochodzące z bazy danych. Właściwie tytuł tego rozdziału powinien brzmieć „Bazy danych, XML i Flash”, ponieważ w takiej kolejności zajmiemy się tymi zagadnieniami.

Bazy danych

Na rynku oprogramowania jest dostępnych wiele różnych programów baz danych, zaś wybór odpowiedniego wiąże się z rozważeniem takich aspektów jak koszty, elastyczność, wydajność i rozszerzalność. Niektóre z nich są również łatwiejsze do opanowania od innych.

Do najbardziej znanych komercyjnych programów baz danych należą Oracle, Microsoft SQL Server i Postgres. Jednak dostępnych jest również mnóstwo „darmowych" programów baz danych, które znakomicie nadają się do większości zastosowań. Generalnie, jeśli zamierzasz korzystać z systemu RDBMS (*Relational DataBase Management System* — system obsługi relacyjnej bazy danych), musi on być zgodny z językiem SQL.

RDBMS to taki system obsługi bazy danych, który przechowuje dane w postaci powiązanych ze sobą tabel. Istnieje jedynie kilka reguł, których należy przestrzegać, by utworzyć dobrą relacyjną bazę danych (dokładniej, jest ich dwanaście (lub trzynaście, w zależności, z kim rozmawiasz); poszukaj hasła *Dr E F Codd* w dobrej wyszukiwarce internetowej). Język SQL (*Structured Query Language* — strukturalny język zapytań) stanowi narzędzie służące do formułowania zapytań dla zgodnych z nim baz danych.

W tym rozdziale użyjemy oprogramowania MySQL, jednego z najpopularniejszych darmowych systemów obsługi baz danych, którego opanowanie jest proste, lecz potrafi on realizować praktycznie dowolne zadania.

Oprogramowanie MySQL możesz pobrać ze strony *www.mysql.com*. W przykładach będziemy korzystać z wersji dla systemu Windows, jednak MySQL jest dostępny również dla systemu MacOS X.

W systemie Windows wystarczy uruchomić program instalacyjny, który instaluje serwer bazy danych i klienta, za pomocą którego można się z nią łączyć. Domyślnie instalator umieszcza oprogramowanie w katalogu *c:\mysql*. Aby nie utrudniać sobie życia, zainstaluj oprogramowanie w tym domyślnym katalogu; jeśli jednak chcesz je zainstalować gdzie indziej, podczas instalacji możesz wybrać katalog docelowy.

Ostatnia dystrybucja MySQL zawiera aż pięć równych serwerów, przy czym wszystkie powinny działać na typowym komputerze PC.

Uruchom program *winmysqladmin.exe* (zawarty w katalogu *c:\mysql\bin*). Powinno się pojawić okno dialogowe z prośbą o podanie nazwy użytkownika i hasła (rysunek 12.1).

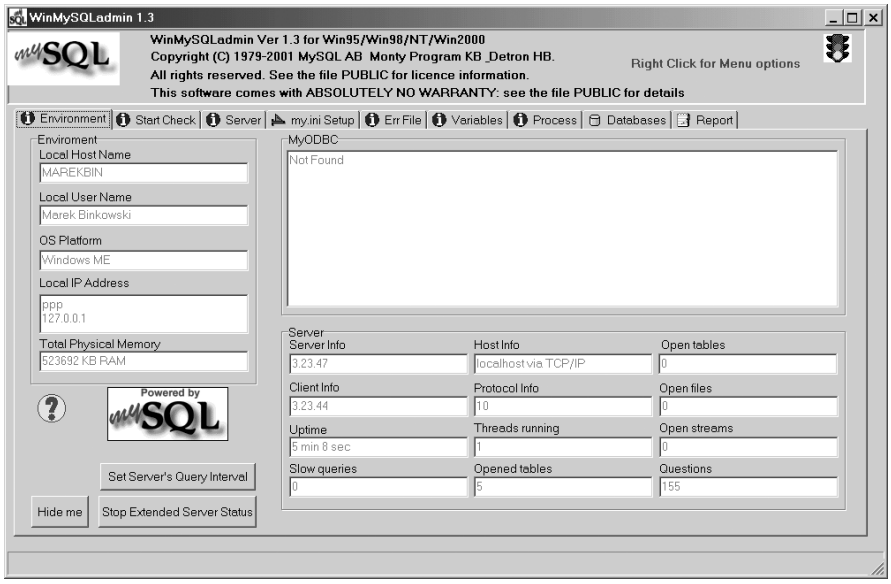
Wpisz nazwę i hasło, którego chcesz używać. Następnie powinno się pojawić główne okno sterowania[1] (rysunek 12.2).

Zielone światło drogowe w prawym górnym narożniku oznacza, że serwer jest uruchomiony i wszystko działa poprawnie.

[1] Jeśli okno to automatycznie się minimalizuje, możesz je wyświetlić, klikając na pasku zadań ikonę świateł drogowych i z menu, które się pojawi, wybierając polecenie *Show Me — przyp. tłum.*

Rysunek 12.1.

Rysunek 12.2.



Bazę danych MySQL, której tu używamy, można podzielić na trzy części:

- Sama baza danych
- Serwer
- Klient

Rzeczywiste bazy danych, z którym będziemy korzystać, muszą być dostępne jednocześnie dla dużej liczby zewnętrznych użytkowników i oferować różne rodzaje połączeń, dlatego stosuje się w nich architekturę klient-serwer. Z bazą możesz łączyć wiele różnych klientów, lecz dystrybucja MySQL jest wyposażona we własnego klienta, którego możesz połączyć z bazą danych MySQL na dowolnym komputerze.

W tym celu musisz określić w kliencie, z jakim serwerem bazy danych chcesz się połączyć, kim jesteś i czy chcesz posługiwać się hasłem dostępu.

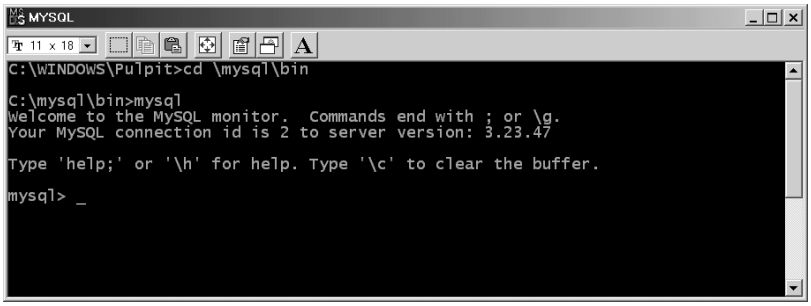
(Jeśli uruchamiasz klienta na tym samym komputerze, na którym jest uruchomiony serwer, nie musisz podawać nazwy komputera macierzystego (*hosta*), choć nie jest to zabronione).

```
C:\mysql\bin> <nazwa bazy danych> -u <NAZWAUŻYTKOWNIKA> - p <hasło> -h <host>
```

Wszystkie te parametry są opcjonalne. Prawdę mówiąc, bezpośrednio po instalacji możesz połączyć się z serwerem, stosując proste polecenie:

```
C:\mysql\bin>mysql
```

Rysunek 12.3.



Po zainstalowaniu oprogramowania MySQL, pliki pomocy w formacie HTML można znaleźć w katalogu *c:\mysql\docs*. Jeśli preferujesz inne formaty dokumentów, odpowiednie pliki pomocy możesz pobrać z witryny MySQL. Ja korzystam głównie z plików PDF. Trzeci rozdział dokumentacji zawiera wprowadzenie do języka SQL (w języku angielskim), wraz z opisem poszczególnych jego elementów.

Poniżej przedstawię podstawowe polecenia SQL, abyśmy mogli rozpocząć pracę.

Pierwsze polecenia SQL

Gdy jesteś już połączony z serwerem, powinieneś się rozejrzeć po otoczeniu. Aby sprawdzić, jakie bazy danych są dostępne, użyj polecenia:

```
mysql> show databases;
```

Każde polecenie SQL w kliencie powinno być zakończone średnikiem `;`. Jeśli po wpisaniu polecenia nie pojawiają się żadne rezultaty, sprawdź, czy wpisałeś średnik (rysunek 12.4).

Rysunek 12.4.

Jak zobaczysz, bezpośrednio po instalacji są dostępne dwie bazy danych: `mysql` oraz `test`. Pierwsza z nich jest odpowiedzialna za wszystkie przywileje dostępu do bazy danych. MySQL korzysta z systemu przywilejów określającego, co może robić konkretny

użytkownik z konkretną bazą danych na danym komputerze macierzystym. Jeśli zamierzasz uruchomić bazę danych w Internecie, powinieneś zapoznać się z regułami przywilejów języka MySQL. Instrukcje, które tu przedstawiamy, pełnią jedynie rolę wprowadzenia.

Jeśli uruchamiasz bazę danych na komputerze, który jest bezpośrednio widoczny w Internecie, pierwszą czynnością powinna być zmiana hasła administratora.

Możesz to zrobić w wierszu poleceń, korzystając z narzędzia `mysqladmin`. Aby zamknąć klienta MySQL, użyj polecenia:

```
mysql> quit
```

Następnie zmień hasło administratora w następujący sposób:

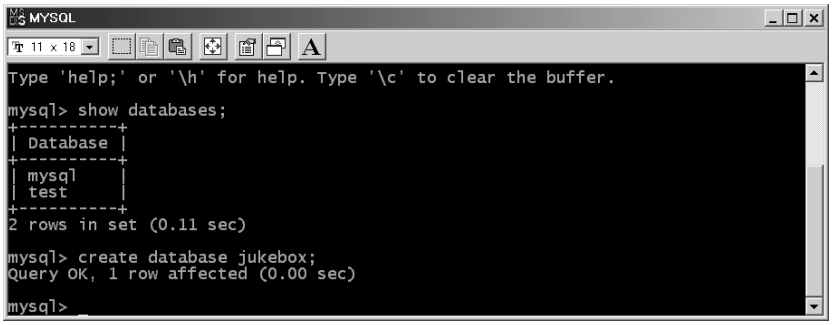
```
C:\mysql\bin> mysqladmin -u root hasło mojehasło
```

Szafa grająca

Utwórzmy nową bazę danych o nazwie `jukebox` (szafa grająca).

Połącz się z serwerem MySQL za pomocą klienta MySQL i wpisz polecenie `create database jukebox;` (rysunek 12.5).

Rysunek 12.5.



Od tej chwili posiadasz nową bazę danych, nie zawierającą na razie żadnych tabel. Utwórzmy tabelę o nazwie `files` (pliki). Tabela będzie zawierała trzy pola — numer identyfikacyjny, pole zawierające nazwę oraz plik zawierający utwory. Musimy określić nazwy poszczególnych pól, typy danych w nich przechowywanych oraz maksymalny rozmiar zawartości każdego pola.

MySQL posiada kilka typów danych, lecz zwykle korzystamy z danych numerycznych lub tekstowych.

`Int`	Liczba całkowita z przedziału od –2147483648 do 2147483647 (jeśli posiada znak)
`Varchar`	Ciąg znaków o zmiennej długości (od 1 do 255 znaków)
`Blob`	(ang. *binary large object*) z tego typu danych korzystamy wówczas, gdy pole ma zawierać więcej niż 255 znaków

Opis pozostałych typów danych znajdziesz w dokumentacji MySQL.

Pierwszemu polu nadamy nazwę `id`. Będzie ono zawierało unikalny klucz identyfikujący każdy rekord. MySQL może automatycznie zwiększać wartość tego identyfikatora, zatem nie musisz się zastanawiać, jaki był identyfikator poprzedniego rekordu.

```
mysql> use jukebox;
mysql> create table files (id int auto_increment not null primary key);
```

Gdy określamy pole jako `not null`, MySQL nie pozwala dodać rekordu do bazy, jeśli to pole nie posiada zawartości. Jest to użyteczne w sytuacji, gdy wypełniasz ważne pola, takie jak adres e-mail, lub gdy zamierzasz za pomocą tego pola powiązać dwie tabele. Lecz nie wyprzedzajmy.

Klient MySQL jest dosyć frustrujący, ponieważ w ścisły sposób sprawdza składnię. To jeden z powodów tworzenia skryptów, które zawierają wszystkie potrzebne polecenia SQL. Inny powód jest taki, że nigdy nie wiesz, kiedy będziesz musiał odtworzyć tabelę.

Aby formułować zapytania, musisz mieć utworzoną bazę danych, na której zapytania będą operowały. Zastosuj poniższą składnię (słowo kluczowe `%path%` zastąp ścieżką, wskazującą plik skryptu MySQL):

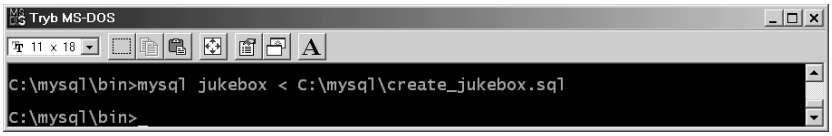
```
mysql nazwa_bazy_danych < %path% mójskrypt.sql
```

W naszym przykładzie umieścimy skrypt *create_jukebox.sql* w katalogu *mysql*, zatem możesz użyć polecenia:

```
C:\mysql\bin>mysql jukebox < C:\mysql\create_jukebox.sql
```

To polecenie spowoduje wykonanie poleceń zawartych w pliku skryptu, przy czym będą one operowały na podanej bazie danych (rysunek 12.6).

Rysunek 12.6.



Oto skrypt SQL, który tworzy tabelę bazy danych `jukebox` i wypełnia ją pewnymi danymi. Skrypt ten mieści się w pliku *create_jukebox.sql*.

```
# To jest skrypt SQL. Tworzenie i używanie skryptów jest wygodne
# ponieważ dosyć często trzeba odtwarzać tabele lub wypełniać je
# danymi testowymi, lub też opróżniać bazę danych
# W skrypcie możesz użyć dowolnych poleceń SQL.

use jukebox;
create table files (id int(7) auto_increment not null primary key);
alter table files add name varchar(40) not null;
alter table files add track varchar(40) not null;

insert into files values (0,'Wayne Krantz', 'the end of wednesday');
insert into files values (0,'Maceo Parker','Pass the peas');
insert into files values (0,'John Martyn','Baby Please come home');
insert into files values (0,'Andy Macdonald','Drum Solo (Rare)');
```

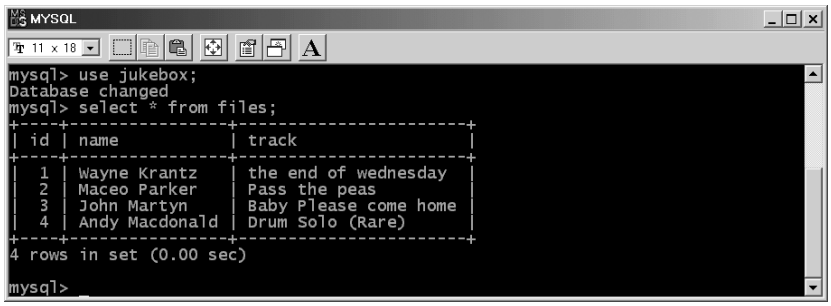
Po utworzeniu bazy danych możesz formułować zapytania SQL, by ją przetestować.

Oto kilka poleceń SQL, z których często korzystam. Wpisz je i zobacz, jaki wynik uzyskasz.

Funkcja select

Rozpocznijmy od takiego prostego zapytania (rysunek 12.7):

Rysunek 12.7.

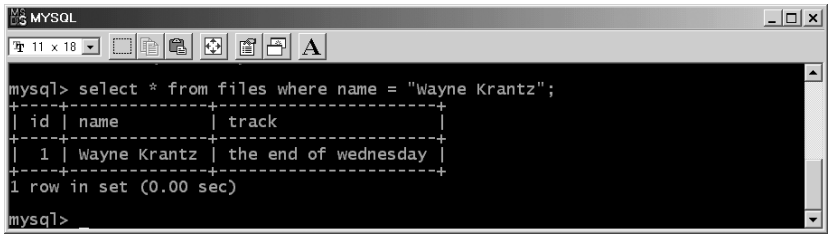



```
select * from files;
```

Jak widać, to zapytanie powoduje wyświetlenie wszystkich rekordów, ponieważ posłużyliśmy się symbolem wieloznacznym *.

Następne zapytanie wybiera z bazy tylko rekord, którego pole `name` zawiera nazwę `Wayne Krantz` (rysunek 12.8).

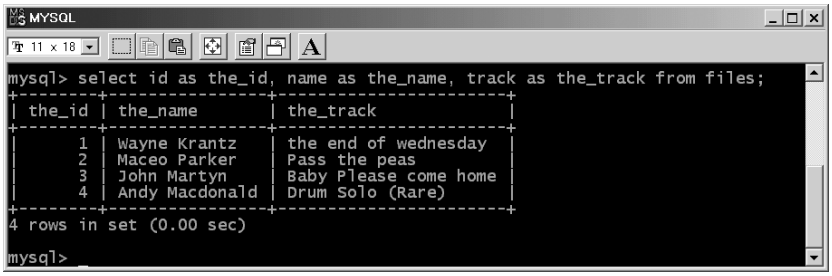
Rysunek 12.8.




```
select * from files where name = "Wayne Krantz";
```

Wreszcie, poniższe zapytanie wyświetla wszystkie rekordy bazy danych, lecz zmienia tytuły pól (rysunek 12.9).

Rysunek 12.9.




```
select id as the_id, name as the_name, track as the_track from files;
```

Funkcja update

Funkcja `update` służy do aktualizacji rekordów w tabeli. Poniższe polecenie we wszystkich rekordach tabeli `files` zmienia zawartość pola `name` na `Wayne Krantz`.

```
update files set name = "Wayne Krantz";
```

To polecenie zamienia wartość pola `name` w rekordzie, którego pole `id` jest równe 2.

```
update files set name = "Wayne Krantz" where id='2';
```

Funkcja delete

Ta funkcja służy do usuwania rekordów z tabeli. Poniższe polecenie usuwa wszystkie rekordy z tabeli `files`.

```
delete from files;
```

Z kolei to polecenie usuwa rekordy, których pole `name` ma wartość `Wayne Krantz`.

```
delete from files where name='Wayne Krantz';
```

Dosyć rzadko używam funkcji `delete` w skryptach uruchamianych w sieci. Wynika to z kilku powodów.

Nie jest bezpiecznym rozwiązaniem udzielanie anonimowemu użytkownikowi przywileju usuwania rekordów. Znacznie bezpieczniejsze jest utworzenie pola, pełniącego funkcję znacznika „aktywny-nieaktywny", który pozwoli użytkownikowi czasowo dezaktywować rekord. Uzyskujemy bardzo podobny efekt, lecz nie wymaga on udzielania użytkownikom tak wielkich przywilejów. Rozpocznij od minimalnych przywilejów i w miarę potrzeb zwiększaj je; nigdy nie postępuj w przeciwnym kierunku. Większość baz danych, które udostępniam, umożliwia anonimowym użytkownikom wybieranie (`select`), wstawianie (`insert`) i aktualizację rekordów (`update`).

Jeśli naprawdę chcesz usuwać rekordy, rób to z bezpiecznego komputera.

Funkcja drop

Tej funkcji używaj z rozwagą. Pozwala ona usuwać tabele, a nawet całe bazy danych. Upewnij się, że wiesz, co robisz, gdy umożliwiasz anonimowym użytkownikom usuwanie danych. Nigdy nie zezwalaj im na usuwanie całych baz danych.

Poniższe polecenie usuwa tabelę `files`.

```
drop table files;
```

Natomiast poniższe polecenie usuwa bazę danych o nazwie `jukebox`, wraz z jej tabelami i ich zawartością.

Zanim przejdziemy do następnego podrozdziału, przyjrzyjmy się jeszcze dwóm poleceniom SQL:

```
grant select, insert, update on jukebox.files to webuser@'%' identified by ¬
'awebpassword';
grant select, insert, update on jukebox.files to webuser@'localhost' identified by ¬
'awebpassword';
```

Powyższe polecenia umożliwią klientowi uruchomionemu na innym komputerze wybieranie (`select`), wstawianie (`insert`) i aktualizację (`update`) rekordów tabeli `files` w bazie danych `jukebox`. Ponadto udostępnią tę bazę danych skryptom uruchamianym z serwera.

Tworzenie skryptów po stronie serwera

Poznałeś już podstawy pracy z klientem MySQL i formułowania zapytań do bazy danych. Możemy zatem przejść do następnego etapu i uruchomić po stronie serwera skrypty, które będą asynchronicznie oddziaływać z bazą danych.

Oznacza to, że będą one zdolne wczytywać materiały do filmu Flasha, a następnie wstawiać pozyskane dane do bazy danych. Jako użytkownik Flasha potrafisz już posługiwać się językami skryptowymi, takimi jak choćby język ActionScript. Jednak jako język skryptów klienta, jest on obarczony pewnymi ograniczeniami, narzuconymi ze względów bezpieczeństwa.

Oto ograniczenia, które nam najbardziej doskwierają:

- Brak dostępu do lokalnych plików (w przeciwnym razie każdy niezaufany program mógłby zapisywać lokalne pliki)
- Brak dostępu do baz danych lub zasobów sieciowych

Skrypty serwerowe, jak się można spodziewać, są uruchamiane na serwerze, a nie na komputerze klienta. W konsekwencji, skrypty serwerowe (generalnie) są zaufane dla aplikacji, która je wywołuje, czyli można przyjąć, że to, co robią, pokrywa się z tym, co zrobiłbyś sam.

Jednym z problemów związanych ze skryptami serwerowymi jest duże obciążenie połączeń, gdy określone zadania muszą być wykonywane na serwerze. Przykładem może być przesyłanie formularza HTML. Gdy użytkownik nie wpisze imienia w odpowiednim polu (co jest wymagane), niekompletny formularz jest przesyłany na serwer, gdzie jest sprawdzany, a następnie odsyłany z powrotem do klienta z prośbą o uzupełnienie brakującego imienia. W rezultacie większość programistów decyduje się na rozwiązania łączące zalety skryptów wykonywanych po stronie klienta i tych działających po stronie serwera (chyba że są oni programistami Java).

Na polu języków serwerowych również masz do dyspozycji bogaty wybór, a ponieważ w większości z nich poradzono już sobie z większością błędów i innych problemów, wybór najczęściej opiera się głównie na osobistych upodobaniach.

Oto najczęściej brane pod uwagę języki programowania skryptów serwerowych:

- ASP (*Active Server Pages* — aktywne strony serwera) — zawierające skrypty w takich językach jak VBScript, CGI Perl, C czy skrypty shell systemu Unix
- Mod-perl
- Python
- Server-Side JavaScript (serwerowy JavaScript)
- PHP
- ColdFusion

Tak bogaty wybór może wywołać konsternację, lecz języki te mają wiele cech wspólnych. Ponadto istnieje wiele narzędzi umożliwiających konwersję skryptów z jednego języka skryptowego na inny. Moim osobistym faworytem jest PHP — język bardzo łatwy do opanowania, z doskonałą obsługą baz danych. Jest on absolutnie darmowy.

W poniższych przykładach posłużymy się językiem PHP.

Jak zainstalować PHP

Najnowszą wersję dystrybucji PHP możesz pobrać z witryny *www.php.net*. Posiada ona wbudowaną obsługę bazy danych MySQL oraz kilku różnych serwerów.

Doskonałym ułatwieniem jest dokumentacja języka PHP, dostępna na wspomnianej witrynie. Zawiera ona komentarze użytkowników na temat wszystkiego, od instalacji po kwestie związane z poszczególnymi funkcjami.

Tworzenie skryptów serwerowych jest nierozerwalnie związane z oprogramowaniem serwera sieciowego, a na tym polu również mamy kilka opcji do wyboru. Uprościmy sobie życie, wybierając oprogramowanie serwerowe najbardziej popularne w Internecie — Apache. Język PHP jest dostępny na swej witrynie w dwóch odmianach, z których jedna jest od razu zabudowana w serwer Apache, jednak jeśli jesteś użytkownikiem Windows, musisz skorzystać z wersji CGI.

Oprogramowanie serwerowe Apache możesz pobrać z witryny *www.apache.org*. Witryna ta jest jednocześnie sercem społeczności, zaangażowanej w wiele innych projektów (w tym bardzo interesujące parsery Java XML). Projekt serwera sieciowego Apache jest znany pod nazwą HTTPD (taka jest nazwa serwera pod systemem Linux).

Jeśli korzystasz z systemu Windows, pobierz binarną wersję dystrybucji win32, którą znajdziesz w podkatalogu *binaries*. Niedawno zaktualizowano instalatory Apache i teraz korzystają one w systemie Windows ze zintegrowanego instalatora. W związku z tym, jeśli korzystasz z jednej ze starszych wersji systemu Windows, być może będziesz musiał pobrać również ostatnią wersję pakietu Windows Installer. Odnośniki do tego pakietu znajdują się na stronie udostępniającej dystrybucje Apache.

W czasie pisania tego tekstu najnowszą wersją serwera Apache była wersja 1.3.22.

Współpraca MySQL, PHP i Apache

Kombinacja tych trzech komponentów zyskuje coraz większą popularność, zaś ich instalacja jest coraz łatwiejsza. Gdy wykonasz poniższe kroki, za pięć minut powinieneś dysponować konfiguracją gotową do pracy. W razie napotkania problemów, poszukaj wskazówek za pomocą wyszukiwarki Google lub innej.

1. Zainstaluj oprogramowanie MySQL. Wymaga to jedynie uruchomienia instalatora i sprawdzenia, czy serwer działa. To zagadnienie już opisaliśmy.

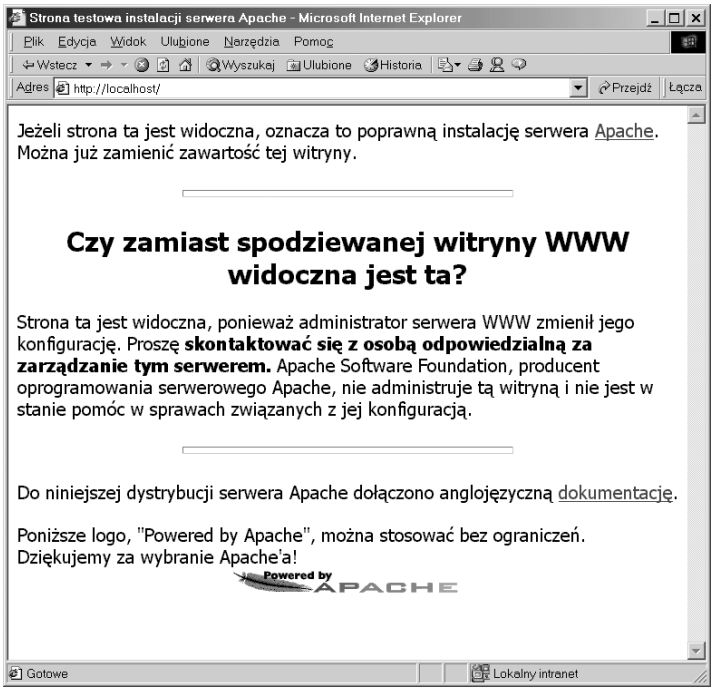
2. Zainstaluj serwer Apache. Tu również wystarczy posłużyć się zwykłym instalatorem Windows.

   W przypadku uruchamiania tego oprogramowania na prawdziwym serwerze internetowym, program Apache poprosi o szczegóły dotyczące domeny serwera, jego nazwy w sieci oraz adresu e-mail administratora.

   Jeśli uruchamiasz ten serwer dla celów treningowych lub testowych, możesz podać dowolną domenę; podaj adres IP 127.0.0.1 (adres lokalnego komputera, gdy nie jest on połączony z siecią) lub twój rzeczywisty adres IP (jeśli chcesz, by inne komputery w sieci również były zdolne do wyświetlania stron). Aby kontynuować, podaj również swój adres e-mail.

   Instalator utworzy skróty w menu *Start* i, jeśli pracujesz w systemie Windows NT lub Windows 2000, zaoferuje możliwość uruchomienia serwera jako serwisu. Na mojej maszynie testowej wolę uruchamiać serwer Apache jako serwis. Możesz również skorzystać ze skrótów, służących do testowania konfiguracji serwera. Uruchom serwer, a następnie użyj adresu *http://localhost/*, *http:///127.0.0.1* lub *http://twój_adres_IP/*, aby zobaczyć testową stronę Apache (rysunek 12.10).

Rysunek 12.10.



Jeżeli łączysz się z Internetem za pomocą modemu, być może konieczna jest zmiana pewnych ustawień usługi *Dial-Up Networking* (DUN), by pojawiła się ta strona. Jeśli komputer próbuje wykręcić numer dostępowy w czasie, gdy chcesz wyświetlić stronę testową Apache, otwórz okno *Internet Options* (Opcje internetowe) (na przykład w przeglądarce Internet Explorer wybierz polecenie *Tools/Internet Explorer* (Narzędzia/Opcje internetowe)) i na zakładce *Connections* (Połączenia) wybierz opcję *Never dial a connection* (Nigdy nie wybieraj połączenia). Gdy sprawdzisz, że strona testowa się pojawia, możesz przywrócić poprzednie ustawienia DUN.

3. Zainstaluj oprogramowanie PHP w katalogu *c:\php*. Jeśli znasz język angielski, możesz przeczytać plik *install.txt* zawarty wewnątrz tego katalogu. Zawiera on informacje na temat instalacji tego oprogramowania.

   Po zainstalowaniu wszystkich plików w katalogu *c:\php* (w naszym przypadku ekran konfiguracji poczty e-mail nie jest istotny) pozostaną do wykonania jeszcze dwie operacje.

   Otwórz plik konfiguracyjny Apache *httpd.conf* (mieści się on w katalogu *C:\Program Files\Apache Group\Apache\conf*) i odszukaj słowo `AddType`. Wstaw poniższe wiersze pomiędzy inne wiersze `AddType`.

   ```
   ScriptAlias /php/ "c:/php/"
   AddType application/x-httpd-php .php
   Action application/x-httpd-php "/php/php.exe"
   ```

   Bardzo ważne — uruchom ponownie serwer Apache. Jeśli pracujesz w systemie Windows NT, uruchom ponownie serwis; w przeciwnym razie przejdź w wierszu poleceń do katalogu Apache i wpisz polecenie:

   ```
   apache -k restart
   ```

4. Utwórz plik o nazwie *php.php* i następującej treści:

   ```
   <?
   phpinfo();
   ?>
   ```

   Zapisz go w głównym katalogu sieciowym serwera Apache (zwykle jest to katalog *C:\Program Files\Apache Group\Apache\htdocs*, chyba że zmieniłeś ścieżkę dostępu podczas instalacji lub też zmodyfikowałeś nazwę katalogu w pliku *httpd.conf*).

   Otwórz przeglądarkę internetową i wpisz adres *http://localhost/php.php*. Powinna się pojawić strona pokazana na rysunku 12.11.

   Strona ta wyświetla wszystkie parametry konfiguracyjne. Jeśli się nie pojawi, wykonaj ponownie operacje związane z instalacją. Jeśli nie zainstalowałeś aplikacji w domyślnych katalogach, spróbuj je przeinstalować — ułatwisz sobie życie.

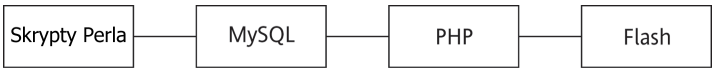
   Po zainstalowaniu tych komponentów masz do dyspozycji metody tworzenia dynamicznych materiałów XML w przeglądarkach i — co bardziej ekscytujące — we Flashu. Możesz uruchamiać skrypty PHP z dowolnego miejsca w głąb głównego katalogu sieciowego (jeśli chcesz, możesz zmienić jego położenie, modyfikując skrypty konfiguracyjne Apache i PHP).

Rysunek 12.11.

Flash

Diagram na rysunku 12.12 ilustruje ogólne zależności pomiędzy poszczególnymi komponentami. Film Flasha odwołuje się do skryptu PHP, który wysyła zapytanie do bazy danych i otrzymuje odpowiedź. Następnie formatuje tę odpowiedź do postaci obiektu XML i przesyła z powrotem do filmu Flasha, który może nimi manipulować lub po prostu je wyświetlić.

Rysunek 12.12.



Często uruchamiam zewnętrzne skrypty, które cyklicznie sprawdzają bazę danych i uruchamiają określone procesy w razie spełnienia określonych warunków przez bazę danych. Na powyższym diagramie są one reprezentowane przez blok „Skrypty Perla”.

Na przykład utworzyłem skrypt, który przenosi wiadomości e-mail z witryny do bazy danych, co pozwala mi następnie w wygodny sposób wysłać potwierdzenia ich otrzymania. Realizuję to za pomocą skryptu Perla, który co jakiś czas sprawdza bazę danych i jeśli znajduje nowe rekordy, automatycznie tworzy wiadomość e-mail i wysyła ją na adres zwrotny. (Jeśli stać cię na licencję, oprogramowanie Microsoft SQL Server umożliwia uruchamianie funkcji VBScript za pomocą detektorów zdarzeń; umożliwiają one wysłanie tych wiadomości e-mail w czasie rzeczywistym).

Dosyć często będziesz tworzył takie łańcuchy procesów i wierz mi, nie zawsze będą one chciały zadziałać od razu. Warto jednak nabrać praktyki, by wiedzieć, co się dzieje na każdym etapie, i co ważniejsze, by potrafić sprawdzić wyniki każdego etapu.

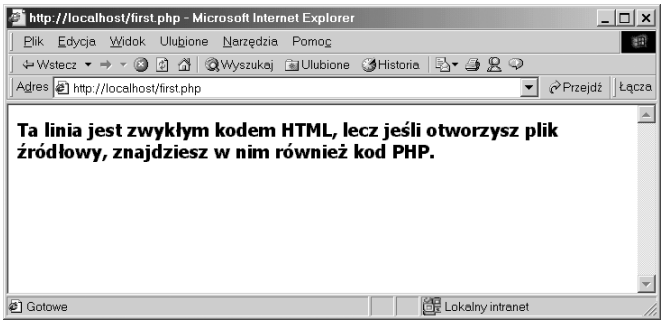
Rozpocznijmy od spojrzenia na etapy działania prostego skryptu PHP. Jeśli wcześniej nie miałeś kontaktu z tym językiem, jednocześnie będzie to dla ciebie pewnego rodzaju

wprowadzenie. Plik ten powinien funkcjonować w głównym katalogu sieciowym. Nie można po prostu „przeciągnąć" pliku PHP do przeglądarki — trzeba odnieść się do niego za pośrednictwem serwera. Na przykład, jeśli plikowi nadasz nazwę *first.php*, odniesiesz się do niego za pomocą adresu *http://localhost.php* (lub na przykład *http://localhost/sites/flash-xml/first.php*, jeśli wcześniej przygotowałeś odpowiednie podkatalogi wewnątrz katalogu sieciowego).

Nasz pierwszy plik PHP

Jeśli pobrałeś i zdekompresowałeś archiwum z przykładami do tej książki, przejdź do katalogu *Rozdzial_12* i odszukaj plik *first.php*. Skopiuj go do głównego katalogu sieciowego (najprawdopodobniej jest to katalog *C:\Program Files\Apache Group\Apache\htdocs*), a następnie otwórz przeglądarkę i odnieś się do tego pliku za pomocą adresu *http://localhost/first.php* (rysunek 12.13).

Rysunek 12.13.



Strona wydaje się banalnie prosta. Spójrz jednak na kod, kryjący się w pliku *first.php*.

```
<?php

//początek sekcji PHP     (to jest komentarz PHP)

/*to
  jest
  wielowierszowy
  komentarz
  php
*/


//    jak widać w tym skrypcie, kody PHP i HTML mogą być dowolnie przemieszane
//    gdy później będziemy uruchamiać skrypty PHP z Flasha, wyniki nie będą w ogóle ¬
      wyświetlane w przeglądarce,
//    natomiast skorzystamy z funkcji sprawdzających poprawność danych ¬
      i wysyłających je z powrotem do Flasha.
//    Tam wykonamy potrzebne obliczenia i wyślemy dane do bazy danych ¬
      (lub pobierzemy je z niej).
//    Skrypt serwera możesz uważać za łącznik pomiędzy komponentami aplikacji.

// koniec kodu PHP. Teraz umieścimy kod HTML. Kod PHP w zasadzie niczego tu nie ¬
   zmienia. Serwer sparsuje tę sekcję
```

```
// i stwierdzi, że zawiera ona jedynie komentarze

?>

<html>
<body>
<P><B>Ten wiersz jest zwykłym kodem HTML, lecz jeśli otworzysz plik źródłowy, ¬
znajdziesz w nim również kod PHP.</B></P>

</body>
</html>
```

Ten przykładowy plik zawiera dwie sekcje. Pierwsza mieści sam kod PHP, zaś druga zwykły kod HTML. Sekcje te w powyższym przykładzie są rozdzielone, lecz kody PHP i HTML mogą być dowolnie przemieszane, pod warunkiem że fragmenty PHP oznaczysz znacznikami <?php oraz php?> (lub po prostu <? oraz ?>).

```
<?php
Echo "Hej, tu jest kod PHP";
php?>
```

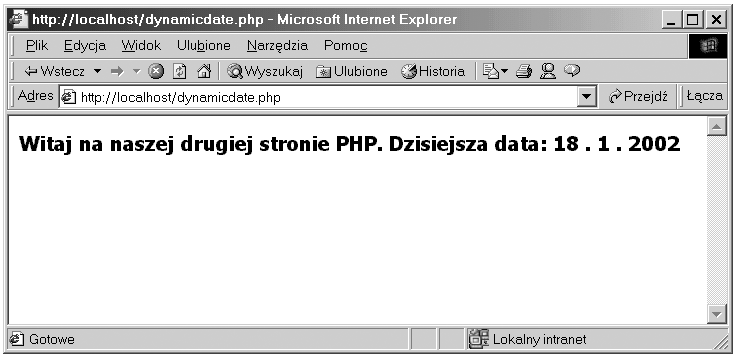
Sekcje PHP są parsowane na serwerze i nie są przesyłane ani wyświetlane na stronie. Możesz zatem używać kodu PHP do generowania dynamicznych elementów, na przykład na dynamicznej stronie HTML. Spójrz na poniższy przykład (znajdziesz go w pliku *dynamicdate.php* — pamiętaj, by skopiować go do głównego katalogu sieciowego).

```
<?php

//W tym przykładzie przemieszamy kody PHP i HTML, by wygenerować aktualną datę ¬
wewnątrz dokumentu

?>

<html>
<body>
<P><B>Witaj na naszej drugiej stronie PHP.  Dzisiejsza data:
<?
$today = getdate();
$month = $today['mon'];
$mday = $today['mday'];
$year = $today['year'];
echo "$mday . $month . $year";
?>
</B></P>

</body>
</html>

<!--funkcja getdate zwraca tablicę zawierającą następujące elementy:

"seconds" - sekundy

"minutes" - minuty

"hours" - godziny

"mday" - dzień miesiąca
```

```
"wday" - dzień tygodnia, numerycznie: od 0 (niedziela) do 6 (sobota)

"mon" - miesiąc, numerycznie

"year" - rok, numerycznie

"yday" - dzień roku, numerycznie; na przykład "299"

"weekday" - dzień tygodnia, tekstowo (pełne, angielskie nazwy dni) ; na przykład ¬
"Friday"

"month" - miesiąc, tekstowo (pełne, angielskie nazwy miesięcy); na przykład "January"

!>
```

Rysunek 12.14 przedstawia stronę wygenerowaną przez powyższy skrypt.

Rysunek 12.14.



Jeśli w przeglądarce wybierzesz polecenie *View/Source* (Widok/Źródło), nie zobaczysz żadnego kodu PHP. Jest to język parsowany przez serwer, działa na serwerze i jedynie rezultaty jego działania są widoczne w przeglądarce. Jedynym sposobem, by poznać, że jest to dynamiczna strona, a nie statyczny tekst, jest rzut oka na rozszerzenie pliku (*.php*).

Wśród przykładowych plików dla rozdziału 12. znajdziesz również plik *variety.php*. Zawiera on przykłady różnych użytecznych rozwiązań wraz z komentarzami. Korzystając z nich oraz dysponując wiedzą na temat języka ActionScript, z łatwością zrozumiesz ich działanie, choć mogą się pojawić pewne nowości w składni.

Tego pliku nie będę tu analizował. Jego zrozumienie nie powinno ci sprawić żadnych trudności — przy okazji przekonasz się, że PHP jest naprawdę łatwym językiem. Uruchom skrypt w przeglądarce i otwórz go w edytorze tekstu. Porównaj rezultat z kodem źródłowym, by sprawdzić, jak działają poszczególne polecenia.

Korzystając z poznanych tu technik, możesz zrealizować większość zadań programistycznych. Pamiętaj, że masz do dyspozycji bogatą dokumentację PHP (w języku angielskim) oraz możesz skorzystać ze wsparcia społeczności programistów PHP.

Więcej informacji na temat języka PHP znajdziesz w książce „Foundation PHP for Flash", którego autorem jest Steve Webster[2].

Łączenie skryptu z bazą danych

Aby używać skryptu w połączeniu z bazą danych, musisz wykonać trzy operacje:

- Połączyć się z bazą danych i wybrać tabelę, na której chcesz operować
- Sformułować zapytania SQL
- Przyjąć rezultaty

Gdy rezultaty zapytań zostaną umieszczone w strukturze danych, możesz nimi manipulować i formatować je w dowolny sposób.

Język PHP posiada różne funkcje służące do łączenia się i formułowania zapytań do różnych typów baz danych, lecz kod SQL jest zawsze ten sam (przynajmniej w większości przypadków).

Oto funkcje MySQL, których będziemy używać:

`mysql_connect`	Otwiera połączenie z serwerem MySQL
`mysql_select_db`	Wybiera bazę danych MySQL
`mysql_query`	Wysyła zapytanie MySQL
`mysql_errno`	Zwraca wartość numeryczną, stanowiącą kod błędu poprzednio wykonywanych operacji MySQL

W dokumentacji znajdziesz wiele innych specjalizowanych funkcji, lecz te cztery zwykle są wystarczające do wykonywania prostych operacji.

Większość problemów ze skryptami serwerowymi i bazami danych dotyczy uprawnień (przywilejów). Wcześniej pokazaliśmy, jak użytkownikowi o nazwie `webuser` przyznać pozwolenie na czytanie danych z bazy `jukebox` za pomocą dowolnego komputera.

```
grant select, insert, update on jukebox.files to webuser@'%' identified by ¬
'awebpassword';
grant select, insert, update on jukebox.files to webuser@'localhost' identified by ¬
'awebpassword';
```

Upewnij się, że twój skrypt posiada uprawnienia dostępu do bazy danych. Pamiętaj, że możesz przyznawać przywileje z poziomu klienta MySQL; powinieneś również sprawdzić, czy użytkownik, któremu przyznałeś przywileje, może się połączyć za pomocą klienta MySQL — powinien to być pierwszy krok w przypadku wystąpienia problemów z działaniem skryptu PHP.

[2] Wydawnictwo Helion wydaje polskie tłumaczenie tej książki pod tytułem „Flash PHP. Podstawy" — *przyp. tłum.*

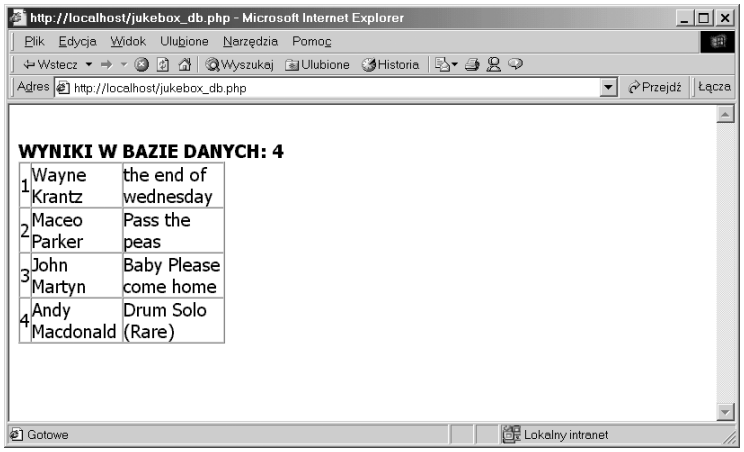
Poniżej przedstawiamy przykład prostego skryptu, który wysyła zapytanie do utworzonej wcześniej bazy danych jukebox i pozyskuje rezultaty. Znajdziesz go w pliku *jukebox_db.php*.

```
<?php
/*
  połącz się z serwerem mysql,
  składnia:  $db=mysql_connect("adres serwera","nazwa użytkownika", "hasło")
*/
$db = mysql_connect("127.0.0.1", "root","");

//wybierz bazę danych lub wyświetl informację o niepowodzeniu
mysql_select_db("jukebox",$db)or die ("&error=problem z wyborem bazy");

//wyślij zapytanie SQL do bazy danych jukebox
$result = mysql_query("SELECT * from files",$db);

//inicjalizacja licznika
$number_of_tracks=0;

//pobierz rezultat zapytania i skonwertuj go na użyteczną tablicę
while ($thisrecord=mysql_fetch_array($result))
{
  /*  oto rekord definiowany bezpośrednio (umieszczany w tablicy):
    $structured_array[0]['id'] holds 1
    $structured_array[0]['name'] holds "artysta"
    $structured_array[0]['track'] holds "utwór"
   gdybyś wywołałał zapytanie select * from jukebox
   w oknie klienta mysql, rekord wyglądałby tak:
     ------------------------------------------------
    |     id        |      name     |     track     |
     ------------------------------------------------
    |      1        |    artysta    |     utwór     |

    dodaj poszczególne rekordy do wspólnej tablicy
  */
  $structured_array[]= array (
          "id"      =>$thisrecord['id'],
              "name"    =>$thisrecord['name'],
              "track"      =>$thisrecord['track']
           );

  //zwiększ wartość licznika, czyli liczbę utworów
  $number_of_tracks++;
}

//przejdź całą tablicę i wyświetl poszczególne rekordy
?>

<html>
<body>
<?
echo "<BR><B><font=\"arial\" size=\"3\">WYNIKI W BAZIE DANYCH: ¬
$number_of_tracks</font>";
echo "</b><table width=\"30%\" border=\"1\" cellspacing=\"0\" cellpadding=\"0\">";

for ($iteration=0;$iteration<$number_of_tracks;$iteration++)
{
```

```
echo "<TR><TD>".$structured_array[$iteration]['id']."</TD><TD>"¬
.$structured_array[$iteration]['name']."</TD><TD>".$structured_array[$iteration] ¬
['track']."</TD></TR>";
}

echo "</table>";
?>
```

Rysunek 12.15 przedstawia rezultat skryptu. Za pomocą kodu PHP utworzyliśmy tabelę HTML, by wyświetlić zawartość bazy w uporządkowany sposób.

Rysunek 12.15.



Choć skrypt zawiera dosyć dużo kodu, jego działanie możemy streścić w prosty sposób.

Pierwsza część skryptu łączy się z bazą danych za pomocą polecenia:

```
$db = mysql_connect("127.0.0.1", "root","");
```

Następnie wybieramy bazę danych, używając polecenia:

```
mysql_select_db("jukebox",$db)or die ("&error=problem z wyborem bazy");
```

Wysyłamy zapytanie, którego rezultat umieszczamy w strukturze tablicowej `$result`:

```
$result = mysql_query("SELECT * from files",$db);
```

Jeśli chciałbyś zobaczyć raport błędu w przypadku problemów z wykonaniem funkcji MySQL, możesz umieścić w skrypcie dodatkowy, poniższy wiersz. Jest on przydatny podczas tworzenia skryptu i pozwala wychwycić przypadkowe błędy, takie jak użycie błędnej nazwy tablicy.

```
echo mysql_errno().": ".mysql_error()."<BR>";
```

Ponadto zwróć uwagę, że przy wysyłaniu zapytania SQL za pomocą kodu PHP nie musisz zakańczać zapytania średnikiem, tak jak w przypadku wpisywania poleceń bezpośrednio w oknie klienta MySQL.

Zadanie, które wydawało się naprawdę skomplikowane, wypełniliśmy w trzech wierszach kodu!

Teraz musimy jeszcze dotrzeć do wyników zapytania.

Funkcja PHP o nazwie `mysql_result()` zwraca zawartość pojedynczej komórki, wchodzącej w skład wyniku zapytania MySQL. Jest ona przydatna na przykład wtedy, gdy wiesz, że rezultat powinien być pojedynczy.

Jednak w większości zastosowań baza danych zwraca więcej komórek wynikowych, udostępnianych przez bardziej przydatną funkcję:

`mysql_fetch_array($result)` — domyślnie funkcja ta udostępnia rezultat w postaci wspólnej tablicy, lecz można też skorzystać z jej użytecznych opcji.

Wspólna tablica umożliwia wygodny dostęp do poszczególnych pól w poszczególnych rekordach.

Nasz przykładowy rezultat zapytania wygląda następująco:

```
1.    Artysta          Utwór
2.    Inny artysta     Inny utwór
3.    Inny artysta     Inny utwór
```

Powyższe dane można przedstawić jako dwuwymiarową strukturę. Możesz odnieść się do konkretnego elementu w tablicy za pomocą indeksu, pamiętając, że tablice są indeksowane od zera, czyli tak:

```
[0][0]   [0][1]   [0][2]
[1][0]   [1][1]   [1][2]
[2][0]   [2][1]   [2][2]
```

W powyższym przykładzie komórka `[0][0]` zawiera wartość `1`, komórka `[0][2]` tekst `Utwór`, natomiast komórka `[2][2]` tekst `Inny utwór`.

Wspólna tablica kojarzy wartości indeksów z rzeczywistymi nazwami, jakie możesz przypisać komórkom. W powyższym przykładzie użyliśmy kombinacji obu rozwiązań.

Wspólną tablicę wyników tworzymy w taki sposób:

```
$structured_array[]= array (
  "id"      =>$thisrecord['id'],
    "nazwa"    =>$thisrecord['name'],
    "utwór"      =>$thisrecord['track']
 );
```

Do poszczególnych komórek odnosimy się wówczas następująco:

```
$structured_array[0]['id']
$structured_array[0]['nazwa']
$structured_array[0]['utwór']
```

Pierwszy indeks służy nam do określania, o który rekord nam chodzi, zaś drugi pozwala odnieść się do konkretnego pola. Jak widzisz, nazwy indeksów nie muszą się pokrywać

z nazwami pól w bazie danych. Oczywiście mogłyby być również takie same, jak w poniższym przykładzie:

```
$structured_array[]= array (
  "id"      =>$thisrecord['id'],
    "name"    =>$thisrecord['name'],
    "track"      =>$thisrecord['track']
 );
```

Wówczas nazwę drugiego artysty z bazy danych odczytałbym w taki sposób:

```
$structured_array[1]['name']
```

Oczywiście dla polskiego czytelnika łatwiejsze będzie posługiwanie się polskimi nazwami pól.

Podsumujmy. Mamy tablicę rekordów, wiemy, ile rezultatów zwraca zapytanie, i znamy słowa kluczowe, za pomocą których możemy odnosić się do poszczególnych pól w rekordach.

Generowanie kodu XML

Rozpocznijmy przykład od prostego dokumentu XML. Znajdziesz go w pliku *jukebox.xml*.

```
<?xml version='1.0' encoding='windows-1250'  ?>

<jukebox>
   <artyści wszyscy="5">
      <artysta>
         <nazwa>Wayne Krantz</nazwa>
         <utwór>The end of wednesday</utwór>
      </artysta>

   </artyści>
</jukebox>
```

Generowanie kodu XML za pomocą skryptu PHP jest dziecinnie proste. Prawdę mówiąc, zwykłe dodanie znaczników <?php php?> na górze powyższego pliku i nadanie mu nazwy *jukebox.php* z technicznego punktu widzenia zmieniłoby go w plik PHP. Jednak bardziej użytecznym rozwiązaniem będzie wygenerowanie każdego wiersza lub fragmentu kodu XML za pomocą polecenia PHP. Tym sposobem możesz decydować o zawartości generowanego kodu XML, umieszczać w skrypcie PHP struktury decyzyjne i inne rozwiązania określające jego postać.

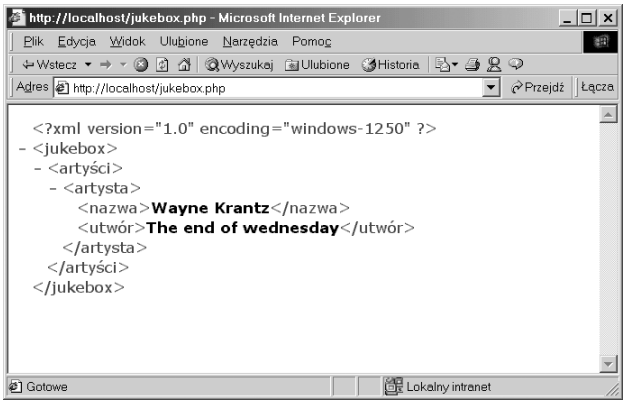
Dlatego nasz plik *jukebox.php* wygląda następująco.

```
<?php
echo "<?xml version=\"1.0\" encoding=\"windows-1250\"?>";
echo "\n\n";


echo "<jukebox>";
echo"<artyści>";
echo"  <artysta>";
```

```
echo"      <nazwa>Wayne Krantz</nazwa>";
echo"    <utwór>The end of wednesday</utwór>";
echo"    </artysta>";

echo"</artyści>";
echo"</jukebox>";

php?>
```

Rysunek 12.16 przedstawia okno przeglądarki po uruchomieniu skryptu *jukebox.php*.

Rysunek 12.16.



Generowanie kodu XML na podstawie bazy danych

Jeśli te same dane byłyby zawarte w bazie danych, moglibyśmy je przekształcić do formatu XML, by następnie wysłać je w tej postaci do klienta, który jest przygotowany do operowania danymi w formacie XML. W tym celu musielibyśmy:

- Sformułować odpowiednie zapytanie SQL
- Wywołać to zapytanie w skrypcie PHP
- Sformatować rezultaty
- Zwrócić (lub przesłać) rezultaty

Przykład skryptu PHP wykonującego powyższe zadania znajdziesz w pliku *jukebox_db_xml.php*. Działa on bardzo podobnie do skryptu *jukebox_db.php*, lecz zamiast tabeli HTML generuje drzewo XML. W obu skryptach korzystamy z utworzonej wcześniej bazy danych `jukebox`. Wiemy, że zawiera ona tabelę `files`, zawierającą rekordy składające się z trzech pól:

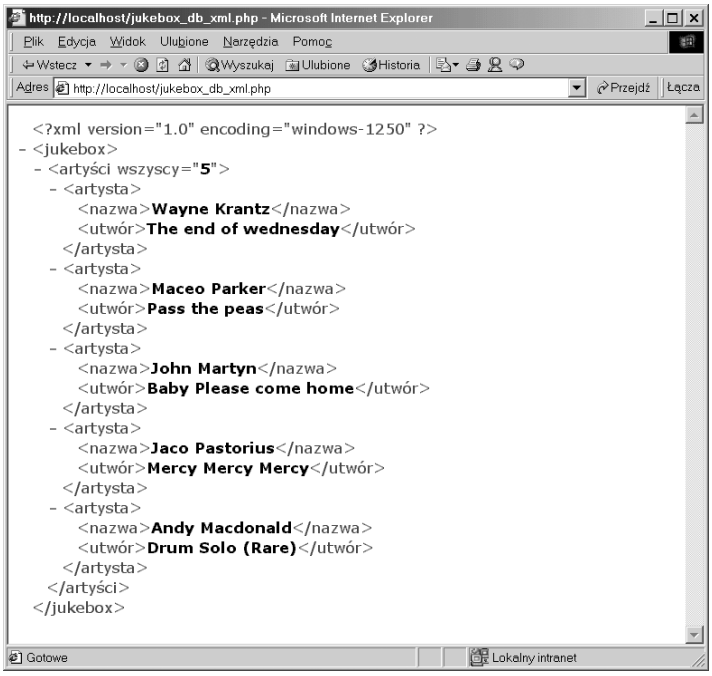
`id`	Automatycznie zwiększana liczba całkowita, stanowiąca identyfikator rekordu
`name`	Wartość typu `varchar`, składająca się maksymalnie z 40 znaków, która nie może być pusta. Zawiera nazwę artysty (zwykle imię i nazwisko)
`track`	Wartość typu `varchar`, składająca się maksymalnie z 40 znaków, która nie może być pusta. Zawiera tytuł utworu

Jak dotrzeć do informacji?

Zobaczmy, jak po stronie klienta odczytamy wygenerowany kod XML. Aby odczytać określony element, konieczne jest przejście od głównego elementu drzewa XML do poziomu, na którym znajduje się ten element. Element główny jest dostępny bezpośrednio i musi on występować w każdym dokumencie XML. Mieści on wszystkie elementy i stanowi punkt odniesienia dla wszelkich „wędrówek" po drzewie.

Zobaczmy drzewo XML, wygenerowane przez skrypt *jukebox_db_xml.php* (rysunek 12.17).

Rysunek 12.17.



(Zwróć uwagę, że ukradkiem dodałem do bazy znakomitego basistę, Jaco Pastoriusa).

W naszym poprzednim przykładzie węzeł `jukebox` miał taką postać:

```
<jukebox>
  …wszystkie inne węzły
</jukebox>
```

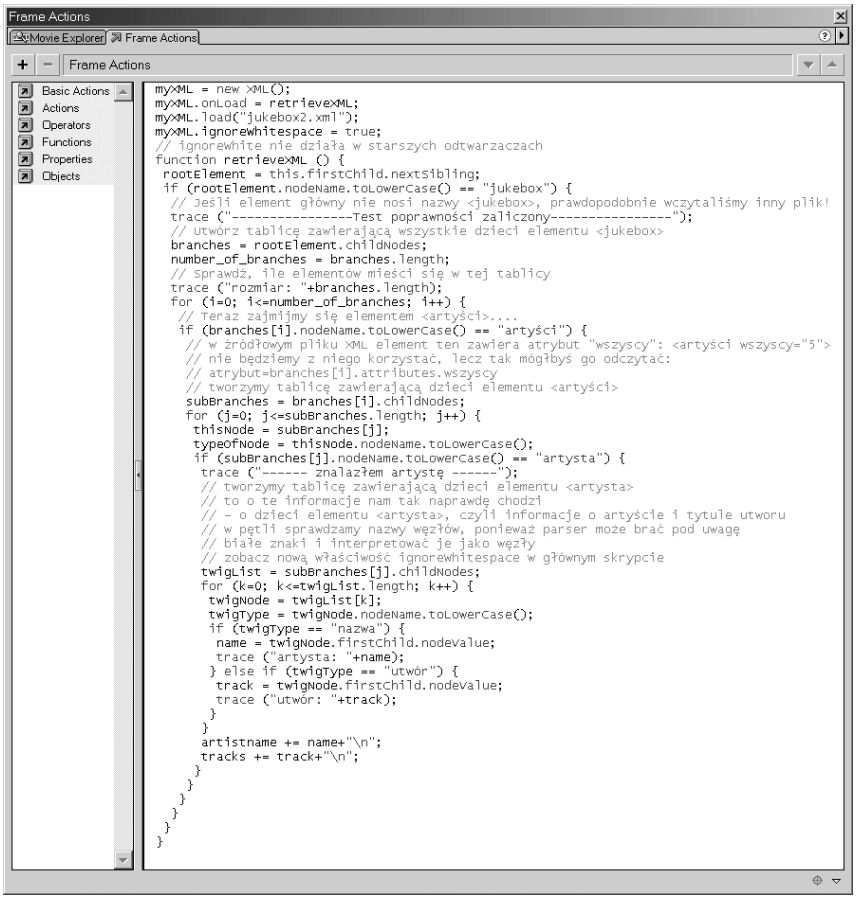
Od tego węzła wyjdziemy w naszym przykładzie. Rysunek 12.18 przedstawia skrypt zawarty w filmie *xmlLoader.fla*.

W filmie Flasha przypisujemy element `<jukebox>` zmiennej `rootElement`, korzystając z polecenia:

```
rootElement = this.firstChild.nextSibling;
```

Element `<jukebox>` jest dzieckiem obiektu *XML*.

Rysunek 12.18.



Jeśli elementem głównym pliku XML nie byłby element <jukebox>, dalsze przetwarzanie pliku nie miałoby sensu, zatem przed przystąpieniem do przetwarzania wykonujemy test poprawności.

Element <jukebox> zawiera dzieci (a raczej jedno dziecko) <artyści>.
Element <artyści> zawiera dzieci <artysta>.
Elementy <artysta> zawierają dzieci <nazwa> oraz <utwór>.

Gdy prześledzisz pętlę, która wyszukuje dzieci elementów, stwierdzisz, że węzłów jest więcej, niż wydaje się być w pliku źródłowym. Wynika to z faktu (który poznaliśmy już wcześniej), że parser XML Flasha interpretuje białe znaki jako węzły. Nowsze wersje odtwarzacza Flash 5 Player obsługują właściwość `ignoreWhitespace`, która pozwala obejść ten problem i zwiększyć wydajność aplikacji (mniej węzłów = mniejsze obciążenie procesora).

```
NazwaObiektu.ignoreWhitespace=true;
```

Czytanie drzewa XML

W tym miejscu bardzo pomocne okazują się właściwości obiektu *XML*, dostępne w języku ActionScript. Ponieważ każdy węzeł w dokumencie XML posiada rodzica i może też posiadać jedno lub więcej dzieci, możesz przejść przez całe drzewo dokumentu, schodząc na kolejne poziomy i odczytując dane w poszczególnych gałęziach.

Tak naprawdę nie wiemy, ile gałęzi, podgałęzi, podpodgałęzi (i tak dalej) może zawierać otrzymany dokument. Dlatego w procesie czytania drzewa posługujemy się zagnieżdżonymi pętlami, w których badamy poszczególne elementy i ich dzieci, ostatecznie docierając do interesującej nas informacji.

W naszym przykładzie na początku tworzymy tablicę `childNodes` zawierającą dzieci elementu `<jukebox>`:

```
branches = rootElement.childNodes;
```

Sprawdzamy długość (liczbę komórek) tej tablicy:

```
number_of_branches = branches.length;
```

Następnie w pętli przeglądamy wszystkie elementy tej tablicy, wyszukując interesujące nas dzieci i rekursywnie schodząc w głąb drzewa.

Na poziomie elementu `<jukebox>` wyszukujemy elementy `<artyści>`; na poziomie elementu `<artyści>` wyszukujemy elementy `<artysta>`; zaś na poziomie elementów `<artysta>` poszukujemy elementów `<nazwa>` i `<ścieżka>`.

W tym momencie osiągamy pojedynczą gałąź i możemy odczytać wartość jej węzła, posługując się właściwością `nodeValue`.

```
name = twigNode.firstChild.nodeValue;
```

Zawartość elementu `<nazwa>Pewna nazwa</nazwa>` jest dostępna jako pierwsze dziecko (`firstChild`) elementu `<name>`.

W ten sposób kończymy wykonywanie wewnętrznej pętli. Wracamy do następnych iteracji zewnętrznych pętli, badających dalsze potomstwo elementu `<jukebox>`.

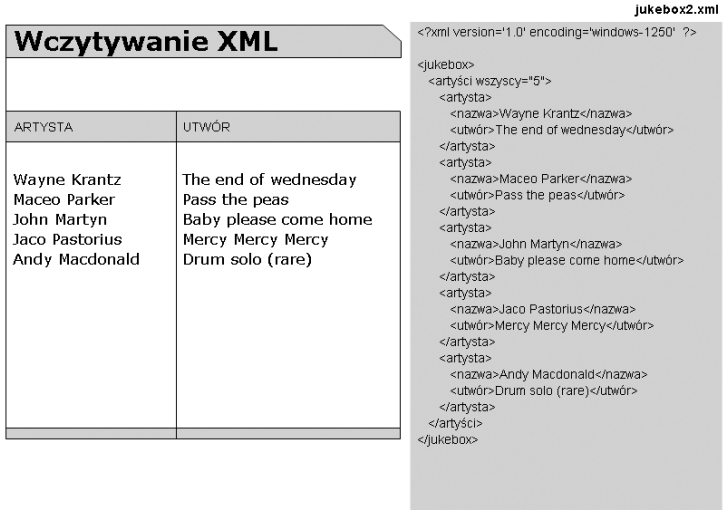
To prowadzi nas do drugiego i następnych elementów `<artysta>` oraz ich dzieci.

Podobne przykłady analizowania dokumentów XML zamieściliśmy we wcześniejszej części książki, zatem zrozumienie tego przykładu nie powinno ci sprawiać żadnych kłopotów.

W ten sposób analizujemy konkretną strukturę dokumentu XML; choć jego zawartość powstała w sposób dynamiczny, sam format pozostaje niezmienny. Jednak język XML jest *rozszerzalny* (ang. *Extensible*), zatem umożliwia tworzenie dowolnych struktur. Już niedługo będziesz tworzył własne struktury i wówczas po stronie klienta będziesz musiał zaadaptować pokazane funkcje tak, aby były w stanie analizować te nowe struktury.

W Internecie możesz znaleźć wiele witryn, które udostępniają źródłowe pliki XML. Warto pobrać sobie jak najwięcej takich plików i spróbować je przeanalizować, by opanować możliwości obsługi dokumentów XML we Flashu i oswoić się z metodami i właściwościami obiektu *XML*. Być może w trakcie takich potyczek przyjdzie ci do głowy nietypowy pomysł na przedstawienie własnych danych i ich obsługę we Flashu. Rysunek 12.19 przedstawia klienta XML, którego utworzyliśmy w tym przykładzie.

Rysunek 12.19.



Rozwiązywanie problemów

Jak wspomnieliśmy wcześniej, współpraca komponentów Flasha, XML, PHP i MySQL wymaga utworzenia łańcucha procesów. Ważne jest, byś w przypadku problemów był w stanie określić, który z komponentów może być ich źródłem. Jeśli aplikacja Flasha nie wyświetla oczekiwanych danych XML, powinieneś po kolei sprawdzić każde ogniwo łańcucha, sprawdzając pośrednie wyniki każdego etapu i starając się wyizolować miejsce, które jest przyczyną problemów.

Istnieje kilka elementów, które powinieneś sprawdzić.

Gdy korzystasz z bazy danych, sprawdź, czy jesteś w stanie połączyć się z nią z klienta MySQL, używając tej samej nazwy użytkownika i hasła, z jakiej korzysta skrypt sieciowy. Pamiętaj, by rzeczywiście połączyć się z bazą danych, używając polecenia `use <baza_danych>`, ponieważ różne bazy danych mogą oferować różne przywileje.

Jeśli to działa, oznacza to, że macierzysty serwer bazy danych działa poprawnie. W idealnym przypadku możesz mieć do dyspozycji inny komputer, połączony z serwerem, na którym wykonasz wszystkie te testy. Jeśli nie jesteś w stanie połączyć się z innego komputera za pomocą danej nazwy użytkownika i hasła, twój skrypt również nie będzie w stanie tego zrobić.

Wskazówka: Sprawdź błędy wyświetlane przez skrypt PHP i umieść w skrypcie *jukebox_db_xml.php* polecenie, które zapamiętuje informacje o błędach, otrzymane z serwera MySQL.

Jeśli skrypt działa poprawnie i otrzymuje informacje z bazy danych, sprawdź, czy formatowanie danych jest poprawne. Spróbuj wczytać stronę w przeglądarce. Czy Internet Explorer wyświetla poprawnie dokument XML, czy też pojawiają się błędy?

Wskazówka: Parser XML wbudowany w przeglądarkę Internet Explorer podświetla wszelkie błędy składniowe. Ponadto za pomocą polecenia *View/Source* (*Widok/Źródło*) możesz zobaczyć w przeglądrce kod generowany przez skrypt PHP. Sprawdź znaczniki domykające i znaki, które powinny wchodzić w skład kodu XML (typowym błędem jest tekst `"Bob & Jane"`; zamiast tego powinieneś użyć `"Bob &amp; Jane"`).

Jeśli wszystko do tej pory funkcjonuje prawidłowo, lecz twoja aplikacja nadal nie działa, umieść akcje `trace` w skrypcie Flasha i przeanalizuj jego działanie.

Jeśli aplikacja nadal nie chce działać, zrób sobie przerwę. Być może zbyt długo wpatrujesz się w to samo i myślisz w ten sam sposób; sześćdziesiąte siódme sprawdzenie tego samego elementu raczej mu już nie pomoże. Być może gdy wrócisz do sprawy wypoczęty, od razu zorientujesz się, co jest nie tak.

Co jeszcze można zrobić z naszą aplikacją

Umieść kilka plików MP3 na lokalnym komputerze i umieść ścieżki dostępu do nich w poszczególnych rekordach bazy danych. Wczytaj te rekordy do klienta Flasha za pomocą skryptu PHP, który sformatował dane do odpowiedniej postaci kodu XML. Za pomocą akcji `getURL()` wczytaj poszczególne pliki MP3 i odtwórz je w aplikacji Flasha.