

Kapitola 23

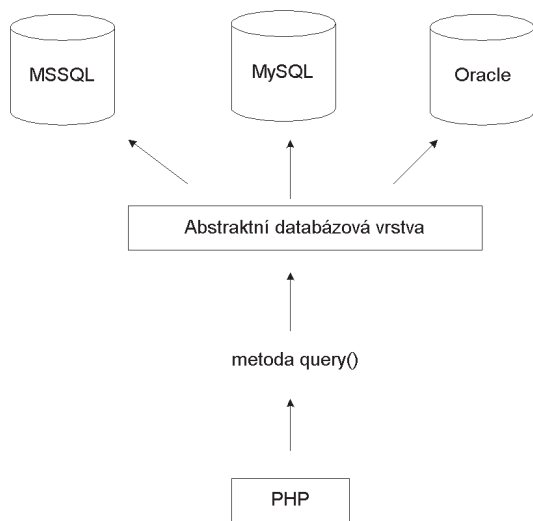
Úvod do PDO

Počet dostupných softwarových řešení je zároveň požehnáním i prokletím. I když je jejich nadbytečná hojnost skvělá pro konečné uživatele, protože si mohou vyhledat takový produkt, který nejlépe vyhovuje jejich konkrétním potřebám, už dlouho je prokázáno, že je to noční můra pro vývojáře a systémové administrátory, protože se od nich požaduje, aby dva nebo více odlišných produktů transparentně vzájemně komunikovaly. I když to, že se ctí různé standardy, jako je třeba XML, značně přispívá v úsilí o interoperabilitu, jsme stále roky vzdáleni od nějakého obecně akceptovatelného řešení.

Tento problém se projeví zvlášť velmi zřetelně tehdy, když aplikace požadují jako datovou základnu nějakou databázi. Zatímco všechny hlavní databáze ctí standard SQL, přestože s různými odchylkami, rozhraní, na nichž jsou závislí programátoři, chtějí-li s databází komunikovat, mohou být značně odlišná (dokonce i tehdy, když jsou dotazy z valné části stejné). Proto bývají aplikace téměř vždy svázané s nějakou konkrétní databází, čímž nutí uživatele, aby si také nainstalovali a udržovali tu hlavní databázi, pokud ji ještě nevlastní. Nebo musejí zvolit alternativní, nějaké patrně méně vyspělé řešení, které bude kompatibilní s jejich stávajícím prostředím. Předpokládejme například, že vaše organizace požaduje aplikaci, která se bude provozovat výlučně na Oracle, ale vaše organizace je přitom standardizovaná na nějakou databázi open-source. Odhodláte se investovat značné prostředky, které bude nutno vložit do nákupu nezbytných licencí Oracle, jste ochotni udržovat tuto databázi jen kvůli tomu, aby se mohla provozovat jedna jediná konkrétní aplikace?

Aby se programátoři velkých korporací nějak s těmito dilematy vypořádali, začali vyvíjet abstraktní databázové vrstvy, které mají sloužit k tomu, aby se mohla oddělit logika aplikace od logiky komunikace s databází. Když se všechny příkazy související s databází mohou prohnat nějakým zevšeobecněným rozhraním, může pak aplikace používat jedno z několika databázových řešení, za předpokladů, že daná databáze podporuje schopnosti, které aplikace požaduje, a že abstraktní vrstva nabízí ovladač kompatibilní s danou databází. Graficky je popisovaný proces znázorněn na obrázku 23-1. Patrně už něco víte o některých rozšířenějších implementacích, z nichž několik je uvedeno v následujícím výčtu:

- **DB:** DB je databázová abstraktní vrstva napsaná v PHP a k dispozici jako balík PEAR. (Další informace o PEAR viz kapitola 11.) V současné době podporuje FrontBase, InterBase, Informix, Mini SQL, MySQL, Oracle, ODBC, PostgreSQL, SQLite a Sybase.



Obrázek 23-1 Oddělení aplikačních a datových vrstev pomocí abstraktní databázové vrstvy

- **JDBC:** jak název implikuje, umožňuje standard Java Database Connectivity (JDBC) programátorům Javy komunikovat s jakoukoli databází, pro kterou je k dispozici ovladač JDBC. Patří sem mj. MSSQL, MySQL, Oracle a PostgreSQL.
- **ODBC:** rozhraní Open Database Connectivity (ODBC) je v současnosti jedna z nejrozšířenějších abstraktních implementací, podporuje ji široká škála aplikací a jazyků, včetně PHP. ovladače ODBC nabízejí všechny hlavní databáze, mezi nimi také ty, které jsou uvedené v odrážce JDBC výše.
- **Perl DBI:** modul Perl Database Interface je standardizovaný prostředek Perlu pro komunikaci s databázemi, a sloužil jako inspirace při tvorbě balíku DB PHP.

Jak vidíte, mají uživatelé PHP po ruce řešení DB i ODBC, proto se může zdát, že jsou vaše potřeby vyřešené, co se týče databázové abstrakce, vyvíjíte-li aplikace poháněné PHP, není-liž pravda? I když jsou tato (a mnohá jiná) řešení hotová a po ruce, už nějakou dobu se vyvíjí ještě lepší řešení, které bylo oficiálně vydané s PHP 5.1. Je známé jako abstraktní vrstva PDO (PHP Data Objects).

Zase další databázová abstraktní vrstva?

Jak PDO v posledních dvou letech dozrával, dost okolo toho huderli vývojáři, kteří buď byli zainteresováni ve vývoji nějakých jiných databázových abstraktních vrstev, nebo patrně byli příliš soustředěni na schopnosti databázové abstraktní vrstvy PDO, než na celou paletu vybavení, které nabízí. Skutečně, PDO poslouží jako ideální náhrada balíku DB a obdobných řešení. PDO je však ve skutečnosti mnohem dál, není to jen pouhá další databázová abstraktní vrstva:

- **Konsistentní kódování:** protože různá databázová rozšíření, která jsou k dispozici pro PHP, psali různí hostující přispěvatelé, neexistuje jednotnost v kódování, navzdory faktu, že všechna tato rozšíření nabízejí v zásadě stejné schopnosti. PDO tuto nejednotnost odstraňuje, protože nabízí jediné rozhraní, které se používá vždy, bez ohledu na to, o jakou databázi se jedná. Navíc

rozšíření je rozděleno do dvou zřetelně vymezených komponent: jádro PDO obsahuje většinu kódu specifického pro PHP, takže se jednotlivé ovladače mohou soustředit výhradně na data. Dále, vývojáři PDO využili svých vědomostí a zkušeností při budování různých databázových rozšíření v posledních letech, takže mohli těžit z toho, co se osvědčilo, to zařadili, a zároveň se pečlivě vystříhali toho, aby zařazovali něco, co se neosvědčilo.

- **Flexibilita:** protože PDO načítá potřebný databázový ovladač až při běhu, není třeba překonfigurovat a překompilovat PHP pokaždé, když se používá jiná databáze. Například potřebujete-li náhle přejít z Oracle na PostgreSQL, prostě načtete ovladač PDO_PGSQL a pracujte (jak se to udělá, o tom více později).
- **Objektově orientované schopnosti:** PDO využívá objektově orientovaných schopností PHP, což vede na vyspělejší a efektivnější databázovou komunikaci.
- **Výkon:** PDO je napsaný v C a vkompilovaný do PHP, což samo o sobě, budou-li všechny ostatní faktory rovnocenné, poskytuje značný nárůst výkonu oproti řešením napsaným v PHP.

Vzhledem k těmto přednostem, proč to nezkusit? V této kapitole se dostatečně obeznámíte s PDO a s myriádami schopností, které nabízí.

Jak se pracuje s PDO

PDO se až pozoruhodně podobá všem databázovým rozšířením, která už dlouhou dobu podporuje PHP; proto, jestliže jste už používali PHP v součinnosti s nějakou databází, bude vám látka prezentovaná v tomto oddílu připadat důvěrně známá. Jak už bylo zmíněno, PDO byl vybudován tak, že jeho tvůrci měli stále na mysli ty nejlepší schopnosti předchozích databázových rozšíření, takže není divu, že v jeho metodách najdete značné podobnosti s tím, co už znáte. Oddíl zahájíme stručným přehledem procesu instalace PDO, pak následuje přehled databázových serverů, které se podporují v současné době. V příkladech kapitoly budeme používat následující tabulku MySQL:

```
CREATE TABLE product (  
    rowid SMALLINT NOT NULL AUTO_INCREMENT,  
    sku CHAR(8) NOT NULL,  
    name VARCHAR(35) NOT NULL,  
    PRIMARY KEY(rowid) );
```

Poznámka překladatele

SKU je zkratka Stock Keeping Unit, v online obchodování jednoznačný identifikátor výrobku na skladě nebo v katalogu

Tabulku naplňte dále uvedenými výrobky:

rowID	SKU	Name
1	ZP457321	Painless Aftershave
2	TY232278	AquaSmooth Toothpaste
3	PO988932	HeadsFree Shampoo
4	KL334899	WhiskerWrecker Razors

Instalace PDO

Jak už bylo zmíněno, je PDO standardně zabalený do PHP 5.1 a novějších verzí, takže provozujete-li tuto verzi, nemusíte podnikat vůbec žádné další kroky. Pracujete-li s nějakou verzí starší než 5.1, i tak můžete pracovat s PDO, když si ho stáhnete z PECL; protože však PDO využívá plně nových objektově orientovaných schopností PHP 5, není možné ho používat v součinnosti s verzí, která je starší než 5.0. Ale ať je to tak či onak, když konfiguruje PHP, musíte přesto explicitně specifikovat ovladače, které chcete zařadit (výjimkou je ovladač SQLITE, který je zařazený standardně). Například chcete-li zapnout podporu ovladače MySQL PDO, přidejte do příkazu `configure` následující přepínač:

```
--with-pdo-mysql=/cesta/k/instalaci/mysql
```

Potřebujete-li se dozvědět víc o jednotlivých ovladačích PDO, vydejte příkaz `configure --help`.

Pracujete-li s PHP 5.1 nebo novější verzí na platformě Windows, tak v době, kdy jsem psal tyto řádky, nebyly ovladače začleněné do distribuce. Proto přejděte na <http://snaps.php.net/win32/>, zadejte patřičný adresář PECL a stáhněte si DLL PDO do adresáře, který máte vyznačený v direktivě `extension_dir` PHP. Pak musíte přidat odkazy na ovladače rozšíření do souboru `php.ini`. Například chcete-li zapnout podporu MySQL, přidejte do sekce `Windows Extensions` řádek:

```
extension=php_pdo_mysql.dll
```

Podpora databází v PDO

Když jsem psal tyto řádky, podporoval PDO devět databází kromě všech těch, které jsou přístupné přes FreeTDS a ODBC:

- **Firebird:** přístupná přes ovladač FIREBIRD.
- **FreeTDS:** to není databáze, ale sada knihoven Unixu, které umožňují programům založených na Unixu hovořit s databázemi MSSQL a Sybase. Přístupná přes ovladač DBLIB.
- **IBM DB2:** přístupná přes ovladač ODBC.
- **Interbase 6:** přístupná přes ovladač FIREBIRD.
- **Microsoft SQL Server:** přístupná přes ovladač MSSQL.
- **MySQL 3.X/4.0:** přístupná přes ovladač MYSQL. Připomínám, že v době, kdy jsem psal tyto řádky, nebylo dostupné rozhraní pro MySQL 5. Protože je jasné, že je to v seznamu priorit vývojářů hodně vysoko, bude to patrně brzy vyřešeno.
- **ODBC v3:** není databáze sama o sobě, ale umožňuje PDO používat v součinnosti s jakoukoli databází kompatibilní s ODBC, která není uvedena v tomto seznamu. Přístupná přes ovladač ODBC.
- **Oracle:** přístupná přes ovladač OCI.
- **PostgreSQL:** přístupná přes ovladač PGSQL.
- **SQLite 3.X:** přístupná přes ovladač SQLITE.
- **Sybase:** přístupná přes ovladač SYBASE.

Tip

Které ovladače PDO máte dostupné ve svém prostředí můžete zjistit tak, že si v prohlížeči zobrazíte výstup z `phpinfo()` a podíváte se do sekce PDO. Nebo zavolejte funkci `pdo_drivers()`, jako zde: `<?php print_r(pdo_drivers()); ?>`.

Připojení k databázovému serveru a výběr databáze

Než můžete začít komunikovat s databází přes PDO, musíte zřídit připojení k serveru a vybrat databázi. Udělá se to konstruktorem PDO. Jeho prototyp vypadá takto:

```
PDO PDO::__construct(string $dsn [, string $username [, string $password  
                        [, array $driver_opts]])
```

Parametr DSN (Data Source Name) se skládá ze dvou prvků: název požadovaného databázového ovladače a všechny nezbytné proměnné databázového připojení, jako jsou název hostitele, port a název databáze. Parametry `username` a `password` specifikují uživatelské jméno a heslo, které se použijí při připojení k databázi. Konečně, pole `driver_opts` specifikuje jakékoli další volby, které by se mohly požadovat nebo jsou žádoucí pro připojení. Seznam dostupných voleb je uveden na konci tohoto oddílu. Konstruktor lze volat několika způsoby, které si teď uvedeme.

Vložíte parametry do konstrukturu

Při prvním způsobu volání konstrukturu PDO se parametry vloží přímo. Například byste ho mohli zavolat třeba takto (jedná se konkrétně o databázi MySQL):

```
$dbh = new PDO("mysql:host=localhost;dbname=corporate", "websiteuser", "secret");
```

Umístíte parametry do souboru

PDO využívá schopnost PHP pracovat s proudy, což otevírá možnost umístit řetězec DSN do separátního souboru, který bude na nějakém místním nebo vzdáleném umístění. Na něj se pak odkážete v konstrukturu, jako zde:

```
$dbh = new PDO("uri:file://usr/local/mysql.dsn");
```

Zajistěte ale, aby soubor vlastnil stejný uživatel, který vykonává skript PHP, a aby tento uživatel měl udělena patřičná přístupová oprávnění.

Odkážete se na soubor php.ini

Je také možné udržovat informace o DSN v souboru `php.ini`, když je přiřadíte do konfiguračního parametru `pdo.dsn.aliasname`, kde `aliasname` je zvolené alias pro DSN, které následně dodáte do konstrukturu. Například v následující ukázce je alias DSN `mysqlpdo`:

```
[PDO]  
pdo.dsn.mysqlpdo = "mysql:dbname=corporate;host=localhost"
```

Alias pak následně uvedete ve volání konstrukturu PDO, jako v:

```
$dbh = new PDO("mysqlpdo", "websiteuser", "secret");
```

Podobně jako v předchozím způsobu, ani tento neumožňuje zařadit do DSN uživatelské jméno a heslo.

Volby PDO vztahující se k připojení

Existuje několik voleb vztahujících se k připojení, jimiž můžete připojení přizpůsobit svým potřebám. Předávají se v poli `driver_opts`. Dostupné volby jsou uvedené v následujícím výčtu:

- **PDO_ATTR_AUTOCOMMIT**: určuje, zda bude PDO potvrzovat změny hned po vykonání každého dotazu, nebo zda bude čekat, až se vykoná metoda `commit()`.
- **PDO_ATTR_CASE**: PDO můžete donutit, aby převáděl získané názvy sloupců na samé velká nebo na samá malá písmena, nebo aby používat názvy sloupců přesně v tom tvaru, v jakém jsou v databázi. Volba se nastavuje na jednu ze tří hodnot: `PDO_CASE_UPPER`, `PDO_CASE_LOWER` a `PDO_CASE_NATURAL`.
- **PDO_ATTR_ERRMODE**: PDO podporuje tři módy oznamování chyb, `PDO_ERRMODE_EXCEPTION`, `PDO_ERRMODE_SILENT` a `PDO_ERRMODE_WARNING`. Určují, za jakých okolností PDO oznámí chybu. Volba se nastavuje na jednu ze tří výše uvedených hodnot, výchozí chování je `PDO_ERRMODE_EXCEPTION`. Tato volba se probírá podrobněji v pozdějším oddílu “Zpracování chyb”.
- **PDO_ATTR_ORACLE_NULLS**: je-li nastavena na `TRUE`, způsobí, že se budou získané prázdné řetězce převádět na `NULL`. Výchozí hodnota je `FALSE`.
- **PDO_ATTR_PERSISTENT**: určuje, zda je připojení trvalé. Výchozí hodnota je `FALSE`.
- **PDO_ATTR_PREFETCH**: jedná se o databázovou schopnost, při které se získává několik řádků, i když klient požaduje v daném okamžiku jen jeden řádek, a to na základě filozofie, že pokud klient požádal o jeden řádek, je pravděpodobné, že bude postupně žádat ještě o další. To snižuje počet požadavků obracejících se na databázi a zvyšuje efektivitu. Volba nastavuje velikost získávané sady v kilobajtech, u těch ovladačů, které tuto schopnost podporují.
- **PDO_ATTR_TIMEOUT**: tato volba nastavuje dobu v sekundách, po kterou se bude čekat, než se skončí.

Další čtyři atributy umožňují dozvědět se dodatečné informace o klientovi, serveru a o stavu připojení. Hodnoty těchto atributů lze získat metodou `getAttribute()`, která se probírá v pozdějším oddílu “Získávání a nastavování atributů”.

- **PDO_ATTR_SERVER_INFO**: obsahuje specifické informace o databázovém serveru. V případě MySQL jsou to data vztahující se k době provozu serveru, kolik bylo celkem dotazů, průměrný počet vykonaných dotazů za sekundu a další důležité informace.
- **PDO_ATTR_SERVER_VERSION**: obsahuje informace o čísle verze databázového serveru.
- **PDO_ATTR_CLIENT_VERSION**: obsahuje informace o čísle verze klienta databáze.
- **PDO_ATTR_CONNECTION_STATUS**: obsahuje informace o stavu připojení k databázi. Například pracujete-li s MySQL, tak po úspěšném připojení atribut obsahuje „localhost via TCP/IP“, zatímco při práci s PostgreSQL obsahuje „Connection OK; waiting to send“.

Jakmile zřídíte připojení, můžete ho začít používat, a to je téma zbývajících částí kapitoly.