

СОДЕРЖАНИЕ

Введение	4
1 Установка PHP расширений для работы с СУБД.....	5
2 Концепция унифицированных функций для доступа к данным	16
3 Доступ к СУБД MYSQL	21
3.1 Функции PHP для работы с СУБД MySQL.....	21
3.2 Создание вспомогательных функций для MySQL	28
3.3 Создание функции MyExecNonQuery.....	30
3.4 Создание функции MyExecQuery.....	33
4 Доступ к СУБД Microsoft SQL Server.....	40
4.1 Функции PHP для доступа к Microsoft SQL.....	40
4.2 Создание вспомогательных функций для Microsoft SQL	47
4.3 Создание функции MsExecNonQuery	48
4.4 Создание функции MsExecQuery	52
5 Доступ к СУБД Firebird.....	59
5.1 Функции PHP для доступа к Firebird	59
5.3 Создание функции FbExecNonQuery	65
5.4 Создание функции FbExecQuery	69
6 Доступ к СУБД PostgreSQL	75
6.1 Функции PHP для доступа к PostgreSQL.....	75
6.2 Создание вспомогательных функций для PostgreSQL	80
6.3 Создание функции PgExecNonQuery	81
6.4 Создание функции PgExecQuery	84
Список литературы.....	90
Приложение 1 "Листинг функций для MySQL"	91
Приложение 2 "Листинг функций для Microsoft SQL Server ".....	97
Приложение 3 "Листинг функций для Firebird"	105
Приложение 4 "Листинг функций для PostgreSQL".....	113

ВВЕДЕНИЕ

В настоящее время информационные системы, используемые в экономике и бизнесе, все чаще строятся в виде Web приложений. В нашей стране для создания таких систем довольно широко используются технологии основанные на PHP. Одной из причин широкого применения PHP является то, что PHP обеспечивает возможность работы с самыми различными СУБД.

В данной работе рассмотрена организация доступа к данным из приложений PHP на примере четырех СУБД: Microsoft SQL Server Express, MySQL Community, PostgreSQL и Firebird. Предложены функции, упрощающие создание информационных систем. Рассмотрен поэтапный процесс их разработки. В приложениях приведены листинги этих функций.

Эти функции можно использовать непосредственно при разработке информационных систем, а также использовать как учебное пособие для разработки собственных средств для работы с базами данных.

Выбор перечня СУБД, рассматриваемых в данной работе, определяется их широким распространением и возможностью их бесплатного применения. Правда СУБД MySQL и Microsoft SQL Server являются проприетарными, но они имеют версии Community и Express, которые дают возможность их использовать для изучения и некоммерческого применения.

1 УСТАНОВКА PHP РАСШИРЕНИЙ ДЛЯ РАБОТЫ С СУБД

На официальном сайте PHP <http://php.net/manual/ru/refs.database.php> можно увидеть, что имеются расширения практически для всех наиболее распространенных СУБД (рис. 1.1).

















- [Расширения для работы с базами данных отдельных производителей](#)
 - [CUBRID](#)
 - [DB++](#)
 - [dBase](#)
 - [filePro](#)
 - [Firebird/InterBase](#)
 - [FrontBase](#)
 - [IBM DB2](#) — IBM DB2, Cloudscape и Apache Derby
 - [Informix](#)
 - [Ingres](#) — Ingres DBMS, EDBC и Enterprise Access Gateways
 - [MaxDB](#)
 - [Mongo](#) — Драйвер MongoDB (устаревший)
 - [MongoDB](#) — Драйвер MongoDB
 - [mSQL](#)
 - [Mssql](#) — Microsoft SQL Server
 - [MySQL](#) — MySQL драйверы и плагины
 - [OCI8](#) — Oracle OCI8
 - [Paradox](#) — Paradox File Access
 - [PostgreSQL](#)
 - [SQLite](#)

Рис. 1.1 — Расширения для работы с СУБД

Эти расширения разработаны различными производителями и имеют свои особенности для каждой конкретной СУБД. Различия между ними определяются также конструктивными возможностями СУБД и языком инструкций для каждой из них.

Перед их использованием для разработки приложений необходимо настроить PHP таким образом, чтобы можно было выполнять действия с требуемыми СУБД. Рассмотрим, как это выполняется.

Скачав PHP с сайта <http://php.net/downloads.php>, можно увидеть, что в папке ext имеются расширения для работы со многими СУБД:

 php_interbase.dll	02.12.2016 13:14
 php_intl.dll	02.12.2016 13:14
 php_ldap.dll	02.12.2016 13:14
 php_mbstring.dll	02.12.2016 13:14
 php_mysqli.dll	02.12.2016 13:14
 php_oci8_12c.dll	02.12.2016 13:14
 php_odbc.dll	02.12.2016 13:14
 php_opcache.dll	02.12.2016 13:14
 php_openssl.dll	02.12.2016 13:14
 php_pdo_firebird.dll	02.12.2016 13:14
 php_pdo_mysql.dll	02.12.2016 13:14
 php_pdo_oci.dll	02.12.2016 13:14
 php_pdo_odbc.dll	02.12.2016 13:14
 php_pdo_pgsql.dll	02.12.2016 13:14
 php_pdo_sqlite.dll	02.12.2016 13:14
 php_pgsql.dll	02.12.2016 13:14

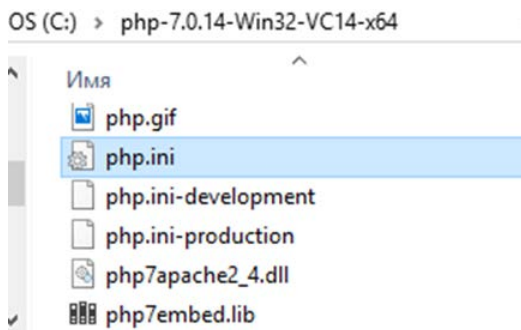
Для работы с интересующими нас СУБД имеются следующие расширения:

- php_interbase.dll для Firebird,
- php_mysqli.dll для MySQL,
- php_pgsql.dll для PostgreSQL.

Кроме вышеперечисленных имеются еще и PDO расширения, например, php_pdo_mysql.dll. Драйверы PDO мы рассматривать не будем — они обеспечивают универсальный механизм доступа для всех СУБД, но при этом падает производительность и нет возможности воспользоваться достоинствами каждой конкретной системы управления базами данных.

Для СУБД Microsoft SQL Server расширения здесь отсутствуют, но их можно скачать с сайта фирмы Microsoft. Это будет рассмотрено позднее в этом же разделе.

Наличие драйверов в папке ext автоматически не обеспечивает возможность работы с СУБД. Надо также внести соответствующую информацию в файл PHP инициализации php.ini:



Открыв этот файл можно увидеть, что первоначально они не активированы:

```
Файл  Правка  Формат  Вид  Справка
;extension=php_ldap.dll
;extension=php_mbstring.dll
;extension=php_exif.dll      ; Must be after mbstring as it depends on it
;extension=php_mysqli.dll
;extension=php_oci8_12c.dll  ; Use with Oracle Database 12c Instant Client
;extension=php_openssl.dll
```

Необходимые расширения следует активировать следующим образом:

```
Файл  Правка  Формат  Вид  Справка
;extension=php_ldap.dll
extension=php_mbstring.dll
;extension=php_exif.dll      ; Must be after mbstring as it depends on it
extension=php_mysqli.dll
;extension=php_oci8_12c.dll  ; Use with Oracle Database 12c Instant Client
;extension=php_openssl.dll
```

Следует обратить внимание, что, кроме этого, активировано расширение для работы со строками, состоящими из многобайтовых символов: `extension=php_mbstring.dll`

Аналогичные действия необходимо выполнить для каждой СУБД, с которой будет работать информационная система.

После изменения в массив `php.ini` требуется перезапустить службу, которая поддерживает WEB сервер (рис. 1.2), например, Apache.

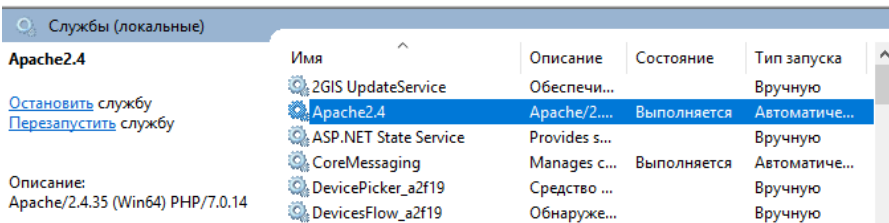


Рис. 1.2 — Перезапуск службы Apache

Для того, чтобы проверить правильность установки, надо запустить в интернет обозревателе приложение `info.php`, которое автоматически устанавливается при установке PHP (рис. 1.3). Если используется WEB сервер Apache, оно находится в папке `htdocs`.

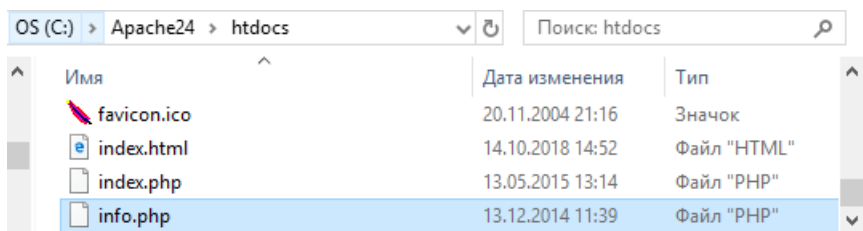


Рис. 1.3 — Приложение `info.php`

В сведениях о составе, установленной версии PHP должна появиться информация приблизительно такого содержания (рис. 1.4–1.7):

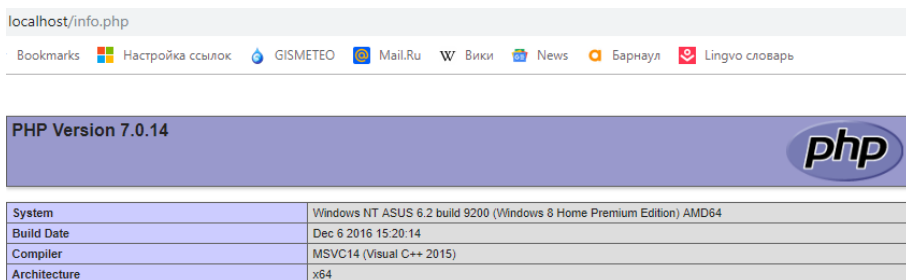


Рис. 1.4 — Заголовок, выдаваемый приложением `info.php`

mysql

Mysql Support		enabled	
Client API library version	mysqlnd 5.0.12-dev - 20150407 - Sld: 241ae00989d1995ffcbf63d579943635fat9972 \$		
Active Persistent Links	0		
Inactive Persistent Links	0		
Active Links	0		

Directive	Local Value	Master Value
mysql.allow_local_infile	On	On
mysql.allow_persistent	On	On
mysql.default_host	<i>no value</i>	<i>no value</i>
mysql.default_port	3306	3306
mysql.default_pw	<i>no value</i>	<i>no value</i>
mysql.default_socket	<i>no value</i>	<i>no value</i>
mysql.default_user	<i>no value</i>	<i>no value</i>
mysql.max_links	Unlimited	Unlimited
mysql.max_persistent	Unlimited	Unlimited
mysql.reconnect	Off	Off
mysql.rollback_on_cached_plink	Off	Off

Рис. 1.5 — Данные по драйверам для MySQL

interbase

Firebird/InterBase Support	dynamic
Compile-time Client Library Version	Firebird API version 25
Run-time Client Library Version	WI-V6.3.3.32900 Firebird 3.0

Directive	Local Value	Master Value
ibase.allow_persistent	On	On
ibase.dateformat	%Y-%m-%d	%Y-%m-%d
ibase.default_charset	<i>no value</i>	<i>no value</i>
ibase.default_db	<i>no value</i>	<i>no value</i>
ibase.default_password	<i>no value</i>	<i>no value</i>
ibase.default_user	<i>no value</i>	<i>no value</i>
ibase.max_links	Unlimited	Unlimited
ibase.max_persistent	Unlimited	Unlimited
ibase.timeformat	%H:%M:%S	%H:%M:%S
ibase.timestampformat	%Y-%m-%d %H:%M:%S	%Y-%m-%d %H:%M:%S

Рис. 1.6 — Данные по драйверам для Firebird

pgsql

PostgreSQL Support		enabled	
PostgreSQL(libpq) Version	9.6.0		
PostgreSQL(libpq)	PostgreSQL 9.6.0 (win32)		
Multibyte character support	enabled		
SSL support	enabled		
Active Persistent Links	0		
Active Links	0		

Directive	Local Value	Master Value
pgsql.allow_persistent	On	On
pgsql.auto_reset_persistent	Off	Off
pgsql.ignore_notice	Off	Off
pgsql.log_notice	Off	Off
pgsql.max_links	Unlimited	Unlimited
pgsql.max_persistent	Unlimited	Unlimited

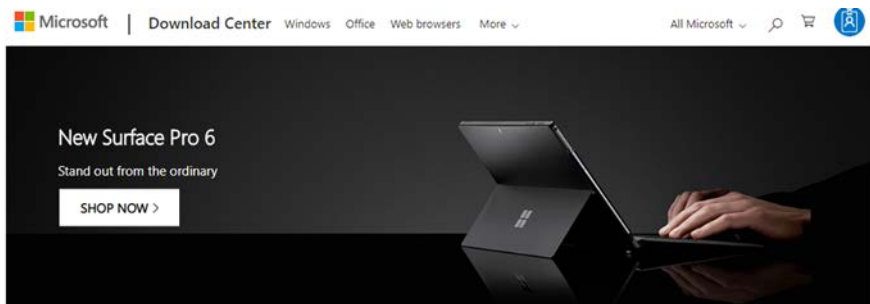
Рис. 1.7 — Данные по драйверам для PostgreSQL

Как видим, установка выполнена и можно начинать работу с этими СУБД.

Описание функций для работы с различными СУБД представлено на сайте <http://php.net/manual/ru/refs.database.php> «Расширения для работы с базами данных».

Для того, чтобы использовать СУБД Microsoft SQL Server, надо получить расширения, созданные разработчиками этой СУБД. Это можно сделать, обратившись к сайту Microsoft (рис. 1.8). Драйверы для СУБД Microsoft SQL Server можно получить по адресу:

<https://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=20098>



Microsoft Drivers for PHP for SQL Server



Рис. 1.8 — Начальная страница для загрузки драйверов для MS SQL Server

После нажатия на кнопку Download необходимо выбрать какой модуль вам необходим.

Для выбора необходимого модуля следует в начале посмотреть, для чего каждый из них предназначен. Эту информацию можно получить, прочитав разъяснения, которые представлены на вышеупомянутом сайте (рис. 1.9, 1.10).

Choose the download you want

<input type="checkbox"/> File Name	Size
<input type="checkbox"/> SQLSRV30.EXE	833 KB
<input type="checkbox"/> Linux_4.0_Install_Instructions.pdf	411 KB
<input type="checkbox"/> SQLSRV31.EXE	489 KB
<input type="checkbox"/> SQLSRV32.EXE	547 KB
<input type="checkbox"/> SQLSRV40.EXE	581 KB

Рис. 1.9 — Перечень модулей для MS SQL Server

Important! Selecting a language below will dynamically change the complete page content to that language.

Language:

The Microsoft Drivers 4.0, 3.2, 3.1, and 3.0 for PHP for SQL Server provide connectivity to Microsoft SQL Server from PHP applications.

Details

System Requirements

Install Instructions

Additional Information

Related Resources

Рис. 1.10 — Дополнительная информация по скачиванию

Раскроем раздел «Системные требования» (System Requirements) (рис. 1.11).



System Requirements

Supported Operating System

Windows 7, Windows 8, Windows 8.1, Windows Server 2008 R2, Windows Server 2008 Service Pack 2, Windows Vista Service Pack 2

Operating Systems supported in this update:

Ubuntu 15.04, 16.04
RedHat 7

Requires PHP 7 or 5.x. For information about how to download and install the latest stable binaries, visit <http://windows.php.net> for more detail.

Version support for PHP is as follows

- Version 4.0 supports PHP 7.0+ on Windows and Linux
- Version 3.2 supports PHP 5.6, 5.5, and 5.4 on Windows
- Version 3.1 supports PHP 5.5 and 5.4 on Windows
- Version 3.0 supports PHP 5.4 on Windows

For more detail and for supported operating systems, see [System Requirements \(Microsoft Drivers for PHP for SQL Server\)](#).

An Internet Information Services (IIS) Web server is required

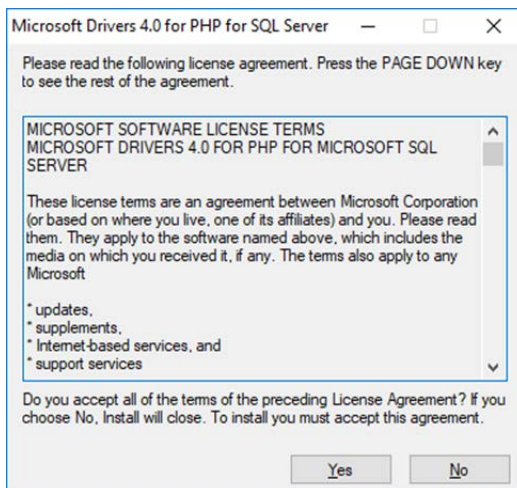
Version 4.0 requires [Microsoft ODBC Driver 11](#) or [Microsoft ODBC Driver 13](#). Version 4.0 for Linux requires [Microsoft ODBC Driver 13](#).

Versions 3.2 and 3.1 of the driver require Microsoft ODBC Driver 11. You can download the [Microsoft ODBC Driver 11 here](#).

Version 3.0 requires the x86 version of Microsoft SQL Server 2012 Native Client.

Рис. 1.11 — Системные требования

Из этого документа можно увидеть, что при использовании PHP 7.0 необходим модуль версии 4.0. Значит необходимо скачать модуль **SQLSRV40.EXE**.



После двойного щелчка по скаченному модулю появляется окно для его установки (рис. 1.12).

Надо нажать кнопку **Yes** и выбрать, куда этот модуль требуется установить (рис. 1.13).

Рис. 1.12 — Лицензионное соглашение

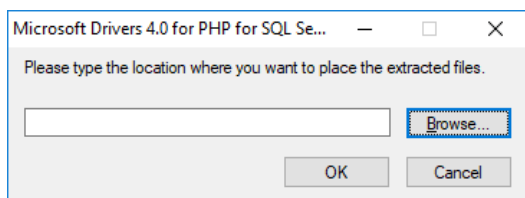


Рис. 1.13 — Выбор места для расположения скачиваемых данных

Расширения для MS Sql Server в конечном итоге необходимо установить в папку ext PHP, но в начале поместим их в какую-нибудь промежуточную папку (рис. 1.14).

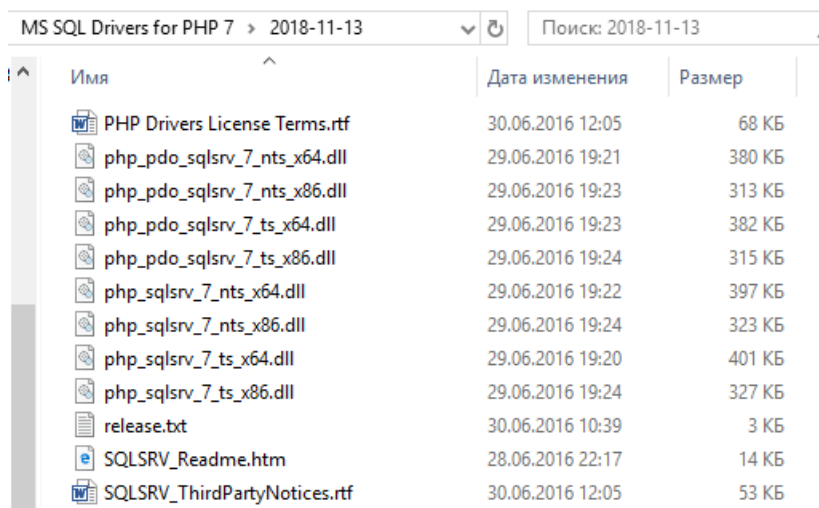


Рис. 1.14 Расширения в промежуточной папке

Как видим, архивный файл содержит большое количество модулей, в том числе имеются файл `SQLSRV_Readme.htm`, в котором содержатся пояснения. Откроем этот файл (рис. 1.15).

В связи с тем, что мы будем делать установку на 64-разрядную систему и PHP у нас работает в потоко-защищенном режиме, переместим модули `php_sqlsrv_7_ts_x64.dll` и `php_pdo_sqlsrv_7_ts_x64.dll` в папку ext.

Ранее я рекомендовал не применять драйверы PDO, но бывают ситуации, когда все-таки их требуется использовать, поэтому мы установили не только основной модуль, но и модуль для режима PDO.

Driver file	PHP version	Thread safe?	Use with PHP .dll
php_sqlsrv_7_nts_x86.dll	7.0	no	php7.dll
php_pdo_sqlsrv_7_nts_x86.dll			
php_sqlsrv_7_ts_x86.dll	7.0	yes	php7ts.dll
php_pdo_sqlsrv_7_ts_x86.dll			
php_sqlsrv_7_nts_x64.dll	7.0	no	php7.dll
php_pdo_sqlsrv_7_nts_x64.dll			
php_sqlsrv_7_ts_x64.dll	7.0	yes	php7ts.dll
php_pdo_sqlsrv_7_ts_x64.dll			

Рис. 1.15 — Сведения о модулях в файле SQLSRV_Readme.htm

Кроме этого, надо открыть файл `php.ini` и добавить в него две строчки:

```
extension=php_sqlsrv_7_ts_x64.dll
extension=php_pdo_sqlsrv_7_ts_x64.dll
```

Теперь следует перезапустить службу Apache (рис. 1.16) для того, чтобы установленные расширения начали работать.

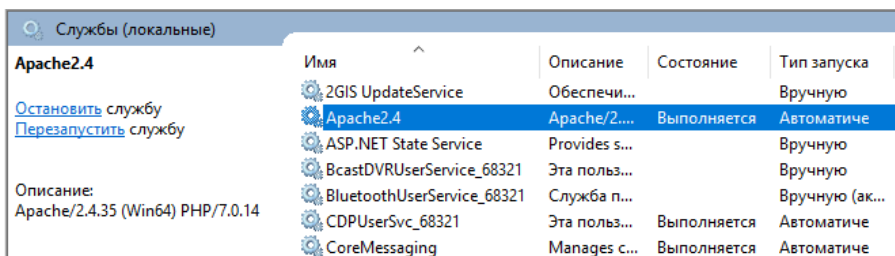


Рис. 1.16 — Перезапуск службы

Для того, чтобы проверить, что установка выполнена, запустим в интернет обозревателе приложение `info.php`, которое находится в папке `htdocs` приложения Apache (рис. 1.17).

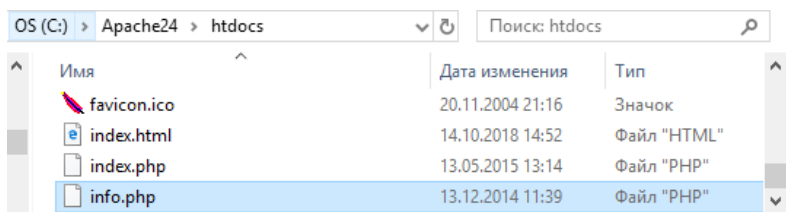


Рис. 1.17 — Запуск приложения info.php

В сведениях о составе, установленной версии PHP должна появиться информация приблизительно такого содержания (рис. 1.18).

localhost/info.php

Bookmarks Настройка ссылок GISMETEO Mail.Ru Вики News Барнаул Lingo словарь

PHP Version 7.0.14

System	Windows NT ASUS 6.2 build 9200 (Windows 8 Home Premium Edition) AMD64	
Build Date	Dec 6 2016 15:20:14	
Compiler	MSVC14 (Visual C++ 2015)	
Architecture	x64	

pdo_sqlsrv

pdo_sqlsrv support	enabled	
ExtensionVer	4.0.8629.2	

Directive	Local Value	Master Value
pdo_sqlsrv.client_buffer_max_kb_size	10240	10240
pdo_sqlsrv.log_severity	0	0

sqlsrv

sqlsrv support	enabled	
ExtensionVer	4.0.8629.2	

Directive	Local Value	Master Value
sqlsrv.ClientBufferMaxKBSize	10240	10240
sqlsrv.LogSeverity	0	0
sqlsrv.LogSubsystems	0	0
sqlsrv.WarningsReturnAsErrors	On	On

Рис. 1.18 Сведения об установленных драйверах

Как видим, установка выполнена и можно начинать работу с Microsoft SQL сервером.

Описание функций, обеспечивающих доступ к СУБД Microsoft SQL Server дано на сайте <http://php.net/manual/ru/book.sqlsrv.php>. Более детальное описание представлено на следующем сайте.

<https://docs.microsoft.com/en-us/sql/connect/php/sqlsrv-driver-api-reference?view=sql-server-2017>

2 КОНЦЕПЦИЯ УНИФИЦИРОВАННЫХ ФУНКЦИЙ ДЛЯ ДОСТУПА К ДАННЫМ

Доступ к данным в информационных системах обычно предназначен для выполнения следующих видов работ:

- получение сведений из базы данных;
- внесение изменений в базу данных.

Иногда требуется выполнять и другие функции, например, создавать таблицы и другие объекты, получать сведения о структуре таблиц (имена, полей типы данных и т. д.). Мы будем рассматривать только два первых вида работ, остальные действия в информационных системах выполняются достаточно редко.

Наша цель — создать унифицированные функции доступа к данным на языке PHP для различных СУБД, причем входные и выходные данные должны быть унифицированными, одинаковыми для всех СУБД. На запросы не должны накладываться ограничения, ведущие к сокращению возможностей, каждой из систем.

При работе с различными СУБД можно увидеть, что функции PHP для доступа к данным могут возвращать информацию в различном виде. Например, расширения для MySQL возвращают информацию о дате в виде текстового поля, а расширения для PostgreSQL в виде объекта типа DateTime. Поэтому определим некоторый единый тип возвращаемых данных.

Функции должны работать таким образом, чтобы можно было с небольшими затратами обеспечить миграцию от одной СУБД к другой. Для этого потребуется только заменить обращение к одной функции на другую и при необходимости в некоторых ситуациях исправить запрос. Без исправления запросов в общем случае не обойтись, так как каждая СУБД использует свой диалект языка SQL.

Например, в качестве знака конкатенации символьных строк в большинстве СУБД используется символ " + ", а в Firebird символ " || ".

Разрабатываемые функции при получении наборов данных должны обеспечивать возможность разделения информации на страницы. При обращении к функции должен задаваться SQL запрос на получение всего набора данных в целом и указываться номер страницы, а функция должна выдавать данные только по одной

странице. Такая возможность существенно упростит разработку приложения, обеспечивающего разбиение данных на страницы.

Для выполнения вышеуказанных действий для каждой СУБД создадим по две функции на языке PHP, которые назовем:

MyExecQuery и MyExecNonQuery для MySQL;

MsExecQuery и MsExecNonQuery для Microsoft SQL;

FbExecQuery и FbExecNonQuery для Firebird;

PgExecQuery и PgExecNonQuery для Postgre SQL;

Каждая из этих функций имеет два параметра — первый из них представляет ассоциативный массив, который содержит сведения, необходимые для открытия соединения с базой данных, второй параметр также является ассоциативным массивом, и содержит SQL запрос и другие сведения, необходимые для выполнения соответствующих действий в базе данных. Результатом выполнения функции будет JSON массив, который содержит сведения, полученные в результате выполнения SQL запроса, а в случае возникновения ошибок — сведения об ошибках.

Первый параметр обеих функций будет содержать следующие элементы:

host — имя сервера;

port — номер порта, является необязательным;

dbName — имя базы данных;

userName — имя пользователя;

password — пароль.

Например, этот параметр может выглядеть следующим образом:

```
[
    'host' => 'localhost',
    'port' => '5432',
    'dbName' => 'okved_db',
    'userName' => 'php',
    'password' => 'phpPassword',
]
```

Для открытия соединений с базой данных могут использоваться и некоторые другие элементы, например `CharacterSet`, который задает кодировку на стороне пользователя. Мы его не включили во входной параметр так как результатом работы будет JSON массив, а для его формирования необходимо чтобы данные

были представлены в кодировке UTF-8. Поэтому при открытии соединения функция будет всегда задавать кодировку UTF-8. В результате этого независимо от кодировки, определенной на стороне сервера, результат всегда будет представлен в UTF-8.

Набор параметров, используемых при соединении с базой данных, не исчерпывается теми, которые здесь перечислены. Например, при работе с СУБД Microsoft SQL Server может использоваться до тридцати различных параметров. В большинстве же случаев достаточно только вышеперечисленных.

Для функций `ExecQuery` второй параметр будет содержать следующие элементы:

- `sqlQuery` — строка запроса;
- `pageSize` — размер страницы (является необязательным);
- `pageNumber` — номер страницы (является необязательным).

Сведения по страницам задаются в связи с тем, что в информационных системах выходные данные часто требуется разбивать на страницы. Каждый запрос при этом будет считывать данные только для одной страницы. Если задан `pageSize`, то должен быть задан и `pageNumber`. Нумерация страниц ведется с единицы.

Этот параметр может быть выглядеть следующим образом:

```
[
  'sqlQuery' => 'SELECT kodokved, nameokved FROM okved',
  'pageSize' => 15,
  'pageNumber' => 5
]
```

Для функций `ExecNonQuery` второй параметр будет содержать следующие элементы:

- `sqlQuery` — строка запроса;
- `newId` — признак, указывающий, что при операции добавления новой записи надо вернуть идентификатор этой записи (является необязательным, определяется только для первичных ключей представляющих из себя столбец с автоувеличением).

Этот параметр может выглядеть следующим образом:

```
[
  'sqlQuery' => "INSERT INTO okved (nameokved) " .
  "VALUES ('Банковская деятельность')",
  'newId' => TRUE
]
```


Результатом выполнения функции типа `ExecQuery` является массив, назовем его условно `result`. Выходная информация должна быть передана в представлении JSON. Преобразование массива в JSON выполняется с помощью PHP функции `json_encode`. Эта функция работает с данными, представленными в коде UTF-8. Результирующий массив `result` содержит два элемента с индексами 0 и 1, которые также являются массивами. Назовем их условно `res` и `data`. Массив `res`, в свою очередь, при нормальном завершении работы, состоит из двух массивов `count` и `names`. Массив `count` содержит 4 элемента с индексами от 0 до 3:

0 — признак, определяющий завершение работы ("OK" — нормальное завершение, "Error" — наличие ошибок),

1 — общее количество записей с данными, которые могут быть получены с помощью заданного SQL запроса (если поддерживается разбиение на страницы, то дается общее количество строк, а не размер страницы),

2 — количество столбцов в одной записи,

3 — номер текущей страницы.

Массив `names` содержит перечень полей массива `data`.

В случае ненормального завершения работы первый элемент массива `res` содержит строковое значение "Error".

Кроме этого, массив `res` может содержать еще несколько элементов, содержащих описание ошибки. Их количество зависит от характера ошибки. Описание ошибки представляет собой группу строк, заданных в массиве `res` начиная с элемента с индексом 1.

При аварийном завершении работы массив `data` отсутствует.

Например, выходной JSON массив при ненормальном завершении работы может иметь следующий вид:

```
[["Error", "Ошибка при обращении к БД", "Access denied"]].
```

В случае нормального завершения работы массив `data` содержит информацию, которую требуется отобразить в виде таблицы. Каждая запись массива `data` является также массивом.

При нормальном завершении результирующий массив может иметь следующий вид:

```
[[["OK", 3, 2,1], ["id", "name"]], [[1, "ИСЭ"], [2, "ИТ"], [3, "ПМ"]]]
```

В приведенном примере массив `data` содержит три записи, каждая из них содержит по два поля.

Результатом выполнения запроса типа `ExecNonQuery` будет массив, который в случае нормального выполнения запроса, содержит три элемента: первый содержит значение "OK", второй — количество записей, которые были затронуты запросом, а третий — идентификатор вновь введенной строки. Третий параметр определяется только в том случае, если была выполнена инструкция `INSERT` и был задан входной параметр `newId` со значением "true".

Например, он может иметь следующий вид:

```
["OK", 1, 235].
```

Если при выполнении функции возникла ошибка, то первый элемент результирующего массива содержит значение "Error", второй и последующие содержат описание ошибки. Например, результирующий массив может иметь следующий вид:

```
["Error", "Запрос не выполнен:", "Unknown column 'id1' in 'where clause'", "update ul_okved set dtend = '2017-03-01' where id1 = 3700;"]
```

Вышеизложенная концепция предполагает, что результат выполнения функции представлен в виде JSON массива. Это связано с тем, что в настоящее время широко используется технология AJAX, которая предполагает обращение к модулям PHP из приложения, выполняемого в интернет обозревателе и написанного на JavaScript. Приложение JavaScript хорошо интегрируется с JSON массивами, ими удобно пользоваться.

В том случае, если приложение не использует технологию AJAX и приложение пишется исключительно на языке PHP, получение данных в виде JSON не требуется. В этом случае в предлагаемых функциях достаточно изменить только последний оператор и информация будет возвращаться в виде PHP массивов. Требуется оператор `return json_encode($res);` заменить на `return $res;`.

3 ДОСТУП К СУБД MYSQL

3.1 Функции PHP для работы с СУБД MySQL

В данном разделе будут рассмотрены следующие функции:

1. `mysqli_connect`
2. `mysqli_connect_errno`
3. `mysqli_connect_error`
4. `mysqli_close`
5. `mysqli_set_charset`
6. `mysqli_query`
7. `mysqli_multi_query`
8. `mysqli_fetch_array`
9. `mysqli_free_result`
10. `mysqli_num_fields`
11. `mysqli_num_rows`
12. `mysqli_fetch_field`
13. `mysqli_affected_rows`
14. `mysqli_insert_id`

Полный перечень функций, используемых для работы СУБД MySQL можно найти на сайте <http://php.net/manual/ru/book.mysql.php>.

3.1.1 Функция, обеспечивающая соединение с базой данных, имеет следующую спецификацию:

```
mysqli_connect ([ string $host =  
ini_get("mysqli.default_host")  
[, string $username= ini_get("mysqli.default_user")  
[, string $passwd = ini_get("mysqli.default_pw")  
[, string $dbname = ""  
[, int $port = ini_get("mysqli.default_port")  
[, string $socket =  
ini_get("mysqli.default_socket") ]]]]] )
```

Параметр `$host` определяет сервер, на котором расположена база данных. Если сервер не задан, то по умолчанию берется сервер, заданный директивой `mysqli.default_host` в файле `php.ini`. Заданное значение может быть именем хоста или IP-адресом.

Передача NULL или строки "localhost" этому параметру означает, что в качестве хоста будет использоваться локальная машина, на которой запущен скрипт.

Параметр `$username` определяет имя пользователя, а `$password` его пароль. Если они не указаны, то используется пользователь и пароль, определенный по умолчанию. В этом случае их значения задаются директивой `mysqli.default_user` в файле `php.ini`. Обычно такая возможность не используется так как с приложениями `php` работает много пользователей с различными правами доступа.

Если после установления соединения будет выполнен второй вызов функции с теми же аргументами, то новое соединение не будет установлено. Вместо этого функция вернёт ссылку на уже установленное соединение.

Если параметр `$username` задан, его значение будет использоваться в качестве имени базы данных по умолчанию при выполнении запросов.

Имя базы данных задается параметром `$dbname`. Если параметр задан, его значение будет использоваться в качестве имени базы данных по умолчанию при выполнении запросов.

Параметр `$port` задает номер порта для подключения к серверу MySQL.

Параметр `$socket` задает сокет или именованный пайп, который необходимо использовать.

3.1.2 Для того, чтобы определить, были ли ошибки при выполнении функции `mysql_connect()` можно использовать функцию `mysqli_connect_errno`, которая имеет следующую спецификацию:

```
int mysqli_connect_errno (void)
```

Она возвращает код ошибки последнего вызова `mysql_connect()`. В случае отсутствия ошибок возвращается 0.

3.1.3 Для того, чтобы получить текстовое описание ошибки используется функция `mysqli_connect_error`, которая имеет следующую спецификацию:

```
string mysqli_connect_error (void)
```

Если ошибка отсутствует она возвращает NULL.

3.1.4 Для закрытия соединения используется функция `mysqli_close`, которая имеет следующую спецификацию:

```
bool mysqli_close (mysqli $link),
```

где `$link` — идентификатор соединения, полученный с помощью `mysqli_connect()`.

Открытые непостоянные соединения MySQL и результирующие наборы автоматически удаляются сразу по окончании работы PHP скрипта. Следовательно, закрывать соединения и очищать результирующие наборы не обязательно, но рекомендуется, так как это сразу же освободит ресурсы базы данных и память, занимаемую результатами выборки, что может положительно сказаться на производительности.

3.1.5 Ввиду того, что функция `mysqli_connect()` не позволяет задать кодировку, которая будет использоваться при получении данных из базы, используется функция `mysqli_set_charset`, которая имеет спецификацию:

```
bool mysqli_set_charset (mysqli $link, string $charset)
```

Параметр `$link` определяет идентификатор соединения, полученный с помощью `mysqli_connect()`.

Параметр `charset` задает идентификатор кодировки, которую требуется установить. Например, при использовании кодировки UTF-8 надо задать значение `'utf8'`. Вставлять тире (`utf-8`) нельзя.

Функция возвращает `TRUE` в случае успешного завершения или `FALSE` при возникновении ошибки.

3.1.6 Функция, обеспечивающая выполнение запроса к базе данных, имеет следующую спецификацию:

```
mixed mysqli_query (mysqli $link, string $query [,  
int $resultmode = MYSQLI_STORE_RESULT ])
```

где `$link` — идентификатор соединения, полученный с помощью функции `mysqli_connect()`.

`$query` — текст запроса.

`$resultmode` — принимает значение `MYSQLI_USE_RESULT`, либо `MYSQLI_STORE_RESULT` в зависимости от требуемого поведения функции. По умолчанию используется `MYSQLI_STORE_RESULT`.

Конец ознакомительного фрагмента.

Приобрести книгу можно

в интернет-магазине

«Электронный универс»

e-Univers.ru