Плагин sqlite

Совместим с «Докувики»

- 2024-02-06 "Kaos" **да**
- 2023-04-04 "Jack Jackrum" **да**
- 2022-07-31 "Igor" **да**
- 2020-07-29 "Hogfather" **да**

Вспомогательный плагин для легкого доступа к базе данных SQLite.

Последнее обновление:
2024-06-25
Предоставляет
Helper
Репозиторий
исходный код

Теги: database, sqlite

Нужен для rating, repository, test

- Andreas Gohr

- logstats
- cosmocode

Этот плагин бесполезен сам по себе. Вместо этого он предоставляет механизмы для других плагинов для создания, управления и доступа к базе данных SQLite. Вам нужно установить его только в том случае, если он требуется другому плагину.

Требования

Этот плагин требует поддержки SQLite и PDO в вашей установке PHP. Убедитесь, что

расширение pdo-sqlite установлено и загружено.

Например, в Debian GNU/Linux обязательно установите php-sqlite3 пакет `sudo apt install php-sqlite3` который предоставит *php-pdo-sqlite* по мере необходимости. После этого выполните перезапуск соответствующего веб-сервера (например, если `sudo systemctl restart apache2` используется Apache HTTP Server) чтобы сделать функциональные возможности доступными для движка DokuWiki.

Загрузка и установка

Найдите и установите плагин с помощью менеджера расширений. Обратитесь к plugins чтобы узнать, как установить плагины вручную.

Изменения

- Version upped (2025/12/02 02:05)
- updated deleted files list (2025/12/01 17:16)
- Version upped (2025/08/27 02:04)
- show database size in admin (2025/08/26 22:37)
- Merge pull request #95 from dokuwiki-translate/lang_update_1127_17477... (2025/05/20 18:28)
- translation update (2025/05/20 11:45)
- Version upped (2025/05/16 02:12)
- make PAGEISHIDDEN return an integer (2025/05/15 11:32)

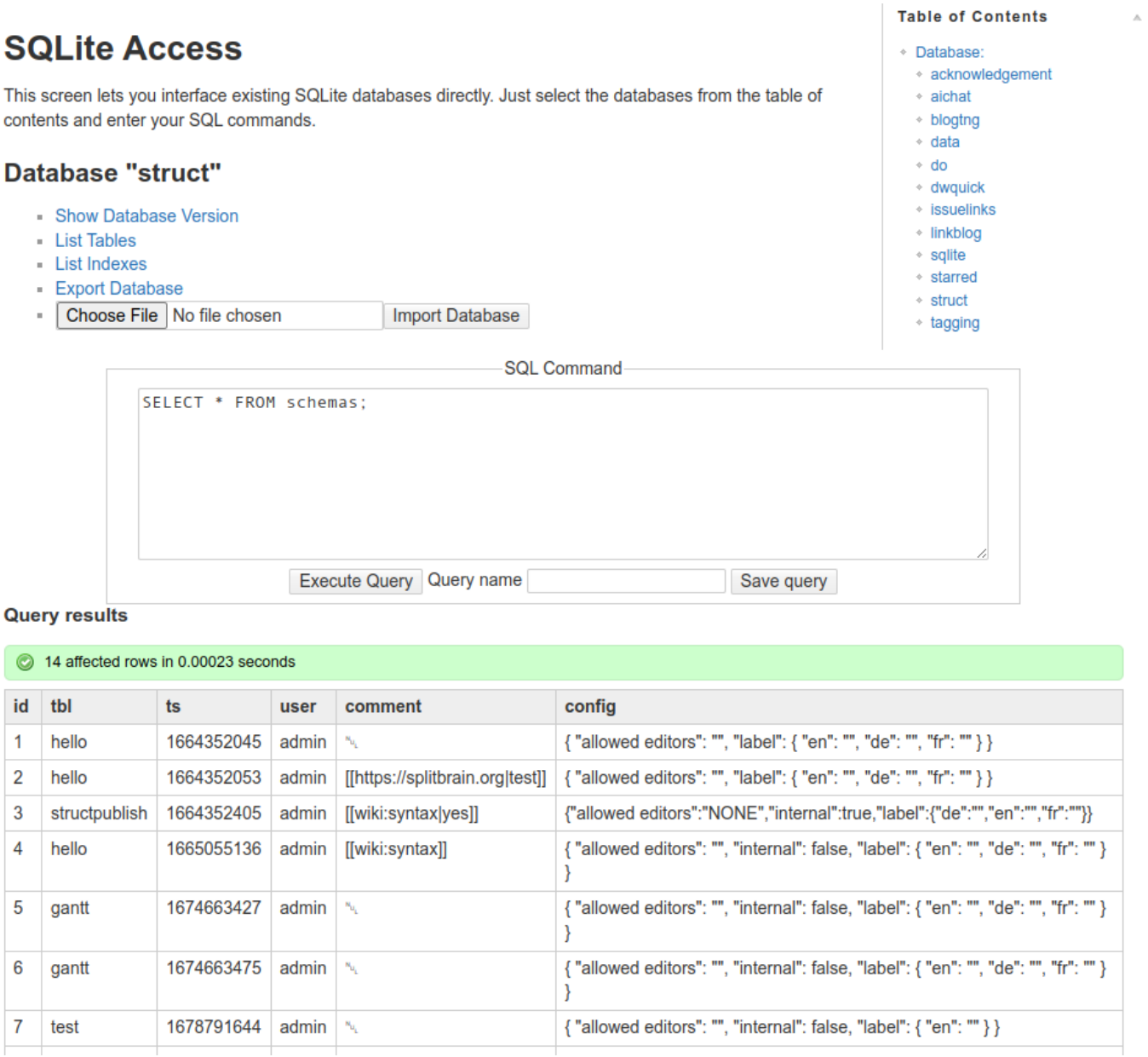
Интерфейс администратора

Плагин поставляется с простым интерфейсом администратора, который дает вам низкоуровневый доступ к любой из доступных баз данных. Это включает:

- выполнение произвольных запросов
- выполнять предопределенные запросы
- экспорт и импорт содержимого базы данных
- сохранять и вызывать самостоятельно определенные запросы

Чтобы использовать любую из его функций, сначала выберите в содержании базу данных, с которой вы хотите работать.


Убедитесь, что вы знаете, что делаете. Этот интерфейс не обеспечивает никаких защитных сеток от удаления или повреждения содержимого вашей базы данных!



id	tbl	ts	user	comment	config
1	hello	1664352045	admin		{ "allowed editors": "", "label": { "en": "", "de": "", "fr": "" } }
2	hello	1664352053	admin	[[https://splitbrain.org|test]]	{ "allowed editors": "", "label": { "en": "", "de": "", "fr": "" } }
3	structpublish	1664352405	admin	[[wiki:syntax|yes]]	{"allowed editors":"NONE","internal":true,"label":{"de":"","en":"","fr":""}}
4	hello	1665055136	admin	[[wiki:syntax]]	{ "allowed editors": "", "internal": false, "label": { "en": "", "de": "", "fr": "" } }
5	gantt	1674663427	admin		{ "allowed editors": "", "internal": false, "label": { "en": "", "de": "", "fr": "" } }
6	gantt	1674663475	admin		{ "allowed editors": "", "internal": false, "label": { "en": "", "de": "", "fr": "" } }
7	test	1678791644	admin		{ "allowed editors": "", "internal": false, "label": { "en": "" } }

Документация разработчика

Этот плагин предоставляет вам все механизмы, необходимые для использования базы данных sqlite в вашем плагине. Вся функциональность предоставляется через `\dokuwiki\plugin\sqlite\SQLiteDB` класс.

Рекомендуется лениво загружать класс, если это необходимо, например, в helper component вашего плагина. Поскольку ваш помощник (обычно) является синглтоном, он гарантирует, что все части вашего кода используют одно и то же подключение к базе данных. А использование функции ленивой загрузки гарантирует, что подключение будет установлено (и инициализировано) только тогда, когда это действительно необходимо.

Ниже вы найдете пример вспомогательного компонента, который поможет вам начать работу.

```
<?php

use dokuwiki\ErrorHandler;
use dokuwiki\Extension\Plugin;
```

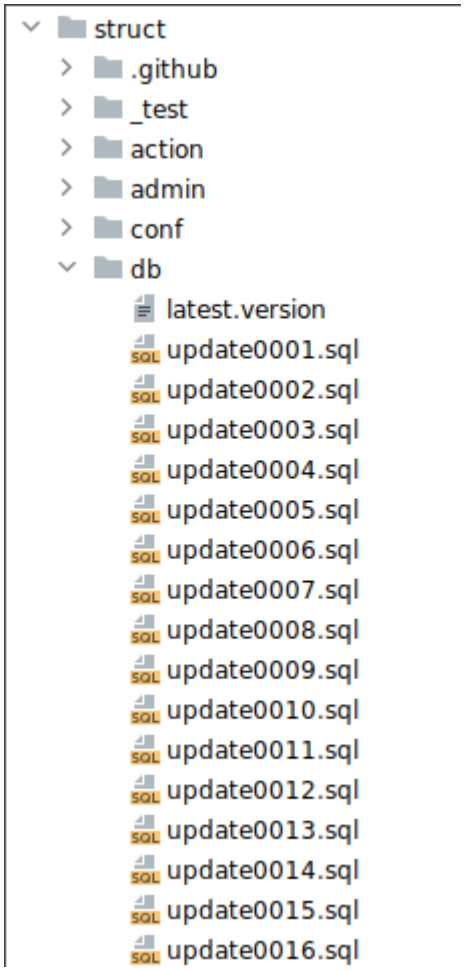
```
use dokuwiki\plugin\sqlite\SQLiteDB;


class helper_plugin_example extends Plugin
{
    protected $db;

    /**
     * Get SQLiteDB instance
     *
     * @return SQLiteDB|null
     */
    public function getDB()
    {
        if ($this->db === null) {
            try {
                $this->db = new SQLiteDB('example', DOKU_PLUGIN .
'example/db/');
            } catch (\Exception $exception) {
                if (defined('DOKU_UNITTEST')) throw new
\RuntimeException('Could not load SQLite', 0, $exception);
                ErrorHandler::logException($exception);
                msg('Couldn\'t load sqlite.', -1);
                return null;
            }
        }
        return $this->db;
    }

    /**
     * @param string $foo
     * @return array
     */
    public function getAllBazForFoo($foo)
    {
        $db = $this->getDB();
        if (!$db) return [];

        $result = $db->queryAll('SELECT id, meta, baz FROM example WHERE foo
= ?', [$foo]);
        return $result;
    }

}
```

Настройка и миграция базы данных

Как вы видели выше, **конструктор** класса **constructor** of the SQLiteDB требует два параметра: **имя базы данных** и **каталог миграции.**

Имя базы данных — это имя файла вашей базы данных SQLite. Если у вас нет очень веской причины, это должно быть имя вашего плагина. В приведенном выше примере имя — , в результате чего создается `example` файл базы данных `.data/meta/example.sqlite3`

Второй параметр определяет каталог, в котором будут находиться файлы миграции базы данных. Большинство плагинов используют `db` каталог в своей структуре плагинов. Используйте константы `DOKU_PLUGIN` или `DIR` для создания соответствующего абсолютного пути.



Итак, что такое каталог миграции? Вашему плагину нужно будет создать схему базы данных и, возможно, заполнить ее некоторыми начальными данными. Когда вы выпускаете новые версии своего плагина, вам может потребоваться обновить схему. Это делается файлами миграции в указанном каталоге. Все это обрабатывается автоматически при вызове конструктора.

Каждый файл соответствует определенной версии базы данных. Версия 1 — это самая первая настройка, которая выполняется при создании базы данных. Каждая последующая версия затем применяется поверх предыдущей.

Номер последней версии должен храниться в файле с именем `latest.version`. Сами файлы обновлений должны именоваться так `updateXXXX.sql`, где XXXX — это 4-значное число, дополненное нулями, начинающееся с `0001`.

Механизм обновления автоматически выполнит все SQL-запросы, найденные в файлах миграции. Выполнение каждого обновления заключено в транзакцию, вам не следует делать это самостоятельно. Если обновление не удается, транзакция откатывается, и обновление отменяется.

Плагин sqlite отслеживает текущую версию базы данных с помощью таблицы с именем `opts`. У вас может не быть таблицы с таким именем!

Примечание: иногда миграции сложнее, чем то, что можно сделать с помощью простых SQL-запросов. Вы можете написать свой собственный код миграции, используя компонент действия и обрабатывая событиеPLUGIN_SQLITE_DATABASE_UPGRADE.

Публичный API

Следующие методы предоставляются классом `SQLiteDB` и могут быть использованы в вашем коде. Обратите внимание, что `PDO::ERRMODE_EXCEPTION` опция PDO включена, и все ошибки будут вызывать исключения. Обязательно обрабатывайте их в вашем коде, где это необходимо.

Список ниже должен дать вам грубый обзор доступных методов. Для получения подробной информации обратитесь к исходному коду и комментариям самого класса: SQLiteDB.php.

> Всегда используйте предоставленные механизмы замены параметров при передаче пользовательских данных в запросы, чтобы избежать атак SQL-инъекций! См. метод query() method.

getPdo()

Этот метод возвращает базовый PDO object осоединения. Вы можете использовать его для вызова всех методов, определенных интерфейсом PDO.

Это полезно, например, для управления транзакциями или регистрации собственных функций.

query()

Базовое выполнение запроса.

Заполнитель в запросе будет заменен данными заменами. Вы можете использовать позиционные ? или именованные `:` заполнители. Этот механизм тот же самый во всех других методах запроса ниже.

Этот метод возвращает объект PDOStatement Не забудьте закрыть его курсор, когда закончите. Обычно вы хотите использовать один из удобных методов ниже.

```
$res = $sqlite->query('SELECT foo FROM example WHERE baz = ?', [$baz]);
$row = $res->fetch();
$res->closeCursor();
```

exec()

Аналогично запросу, но используется для выполнения запросов на изменение данных. Возвращает последний идентификатор вставки в операторах INSERT или количество

затронутых строк в противном случае.

```
$id = $sqlite->exec('INSERT INTO example (bar) VALUES (?)', [$bar]);
echo "new ID is $id";
```

queryAll()

Удобный метод для выполнения запроса и возврата всех строк в виде ассоциативных массивов. Помните об ограничениях памяти при запросе больших наборов данных!

```
$rows = $sqlite->queryAll('SELECT * FROM example WHERE bar = ?', [$bar]);
foreach($rows as $row) {
    foreach($row as $col => $val) {
        echo "$col: $val\n";
    }
    echo "------\n";
}
```

queryRecord()

Удобный метод выполнения запроса и возврата только одной строки.

```
$row = $sqlite->queryRecord('SELECT * FROM example WHERE bar = ?', [$bar]);
foreach($row as $col => $val) {
    echo "$col: $val\n";
}
```

saveRecord()

Удобный метод для вставки или замены одной записи в таблице. Функция ожидает ассоциативный массив для параметра $data, где ключи массива соответствуют именам столбцов.

Если предоставленные данные нарушают ограничение, они либо игнорируются, либо существующие данные заменяются.

Примечание: этот метод обеспечивает только функциональность `INSERT OR IGNORE`. Если вам нужна функциональность UPSERT, создайте свой запрос и используйте exec(). `INSERT OR REPLACE`

```
$data = [
    'foo' => 'Florb',
    'baz' => 12932
];
$new = $sqlite->saveRecord('example', $data);
var_dump($new);
```

queryValue()

Удобный метод запроса одного значения из базы данных. Он вернет первое значение в первой строке набора результатов.

```
$foo = $sqlite->queryValue('SELECT foo FROM example WHERE bar = ?', [$bar]);
echo $foo;
```

queryKeyValueList()

Удобный метод запроса списка ключей и значений из базы данных. Первое значение в каждой строке результата становится ключом, второе значение — значением. Вам нужно самостоятельно позаботиться об уникальности ключей, иначе более поздние значения перезапишут более ранние.

```
$res = $sqlite->queryKeyValueList('SELECT foo,baz FROM example WHERE bar =
?', [$bar]);
foreach($res as $key => $val) {
    echo "$key: $val\n";
}
```

Расширения SQL

SQLite имеет уникальную функцию, которая заключается в том, что PHP-код может быть зарегистрирован как собственные функции SQL, которые затем могут использоваться в ваших запросах. Плагин sqlite регистрирует несколько пользовательских функций, описанных ниже.

Подробную информацию смотрите в классе Functions:Functions.php.

Ваш плагин может регистрировать собственные функции с помощью `getPDO()->sqliteCreateFunction()` и `getPDO()->sqliteCreateAggregate()`.

ПОЛУЧИТЬ УРОВЕНЬ ДОСТУПА

Возвращает целочисленный уровень разрешений «ДокуВики» для заданного идентификатора страницы.

Примечание: эта функция также вернет AUTH_NONE для скрытых страниц.

```
$sqlite->queryAll(
  'SELECT page, foo, bar FROM example WHERE GETACCESSLEVEL(page) > ?',
  [AUTH_READ]
);
```

СТРАНИЦА СУЩЕСТВУЕТ

Проверяет, существует ли заданное имя страницы. Возвращает 1 для существующих страниц, 0 в противном случае.

```
$sqlite->queryAll(
  'SELECT page, foo, bar FROM example WHERE PAGEEXISTS(page) = 1'
);
```

РЕГ. ВЫРАЖЕНИЕ

Сопоставляет значение с заданным регулярным выражением. Выражение используется с / разделителями и u модификатором Unicode.

```
$sqlite->queryAll(
  'SELECT page, foo, bar FROM example WHERE REGEXP("^wiki:.*", page)'
);
```

RESOLVEPAGE

Разрешает заданную относительную страницу по заданной контекстной странице.

```
$sqlite->queryAll(
  'SELECT RESOLVEPAGE(page, ?), foo, bar FROM example',
  [$context]
);
```

GROUP_CONCAT_DISTINCT

Это агрегатная функция, которая работает как встроенная функция GROUP_CONCAT Однако она гарантирует, что будут возвращены только уникальные значения.

Примечание: по возможности следует использовать собственный метод. К сожалению, собственный метод не поддерживает ключевое слово DISTINCT при использовании пользовательского разделителя. Только если другие варианты невозможны, используйте эту зарегистрированную функцию.

```
$sqlite->queryAll(
  'SELECT foo, GROUP_CONCAT_DISTINCT(bar,' ') FROM example GROUP BY foo',
  [$context]
);
```

События

Плагин будет инициировать следующие события, на которые ваш плагин может

зарегистрироваться.

PLUGIN_SQLITE_DATABASE_UPGRADE

Это событие запускается при применении любой новой миграции (см. выше). Это позволяет выполнять более сложные миграции, требующие больше, чем просто SQL.

Предоставленные данные выглядят следующим образом:

```
$data = [
    'database' => 'example', // the database name
    'from' => 5, // previous version
    'to' => 6, // version this migration migrates to
    'file' => /some/path/lib/plugin/example/db/update0006.sql, // the
associated update file
    'sqlite' => [object] // deprecated instance of the old helper, do not
use
    'adapter' => SQLiteDB // instance of the SQLiteDB class
];
```


При обработке этого события обязательно убедитесь, что вы обрабатываете только обновление своей собственной базы данных.

Вот пример для начала:

```
public function handle_migrations(Doku_Event $event, $param) {
    // replace example with name of your plugin
    if ($event->data['database'] !== 'example') {
        return;
    }

    // code to handle update to version 7
    if($data['to'] === 7) {
        $data['adapter']->exec('....');
        $event->result = true;
    }
}
```

Событие BEFORE выполняется до применения файла миграции. Действие по умолчанию — применить этот файл и увеличить `dbversion` поле в `opts` таблице.

Когда событие предотвращает значение по умолчанию, `$event→result` необходимо установить значение! Если результат истинный, `dbversion` увеличивается и запускаются любые дополнительные миграции. Если результат ложный, процесс миграции прерывается.

Событие BEFORE может опционально изменить `file` атрибут данных для загрузки другого файла.

Событие AFTER срабатывает только после применения файла миграции или после того,

BEFORE как событие установило истинный результат. В то AFTER время `dbversion` уже был увеличен.

Код, связанный с этим событием, используемый в любых файлах плагинов и шаблонов

PLUGIN_SQLITE_QUERY_SAVE

Это событие срабатывает при сохранении запроса в интерфейсе администратора.

Вы можете предотвратить сохранение в событии BEFORE или изменить имя или запрос.

```
$event->data = [
    'sqlitedb' => SQLiteDB, // an instance of the SQLiteDB class for access to the ''sqlite'' database.
    'upstream' => 'example', // the name of the selected database, for which the query is stored
    'name' => &$name, // The name the user picked for this query
    'query' => &$query, // The SQL for this query
]
```

Код, связанный с этим событием, используемый в любых файлах плагинов и шаблонов

PLUGIN_SQLITE_QUERY_DELETE

Это событие срабатывает при удалении запроса в интерфейсе администратора.

Вы можете предотвратить удаление в событии BEFORE или изменить имя (это приведет к удалению другого запроса).

```
$event->data = [
    'sqlitedb' => SQLiteDB, // an instance of the SQLiteDB class for access to the ''sqlite'' database.
    'upstream' => 'example', // the name of the selected database, for which the query is stored
    'name' => &$name, // The name the user picked for this query
]
```

Код, связанный с этим событием, используемый в любых файлах плагинов и шаблонов

Журнал медленных запросов

Если включено ведение журнала отладки (см. параметр dontlog запросы, которые занимают больше 0,2 секунд, будут регистрироваться как медленные. Используйте это для оптимизации и отладки ваших запросов.

Изменения по сравнению с предыдущими версиями

В июле 2023 года была выпущена полностью переработанная версия, которая ввела `SQLiteDB` класс в пользу `helper_plugin_sqlite`. Последний по-прежнему предоставляется и внутренне использует `SQLiteDB` класс. Это означает, что новый плагин должен быть полностью обратно совместим со старыми плагинами.

Обсуждения в соответствующем Pull Request могут оказаться поучительными.

Однако некоторые мелочи могут работать немного по-другому. Самое важное, что ранее зарегистрированный метод GROUP_CONCAT был переименован в GROUP_CONCAT_DISTINCT. Пожалуйста, ознакомьтесь с егодокументацией выше и при необходимости настройте свой плагин.

Разработчикам плагинов рекомендуется обновить свои плагины, чтобы использовать новый, более чистый интерфейс SQLiteDB. Помощник теперь устарел.

Дополнения и Файлы

Плагин sqlite