

# DOKUMENTACE TOTELIB 1.0

---

Autor: Josef Dohnal  
Revize – verze: 20190612

## Obsah

1	Popis	3
2	Licence	3
3	Instalace	3
3.1	Composer .....	3
3.2	Stažení z GitHub.....	3
4	Použití	3
4.1	Konfigurace.....	4
4.2	Základní použití knihovny.....	4
4.3	Rozšířené možnosti knihovny.....	5
4.4	Pomocné funkce .....	6
4.5	PDO .....	6
4.6	Adminer .....	7
5	Dokumentace všech funkcí	7

## 1 Popis

Knihovna ToteLib je určena k vytváření spojení do SQL databáze TOTE pro firmu Personna International CZ s.r.o.

Knihovna je vytvořena pro PHP verze 7.2 a vyšší.

Knihovna zapouzdřuje načtení přihlašovacích údajů z .env souboru, vytvoření spojení s databází, základní funkce pro práci s databází a možnost procházet databázi ve webovém prohlížeči.

Knihovna je umístěna na GitHubu na adrese <https://github.com/Mepatek/totelib>

Knihovna pro svůj chod vyžaduje další knihovny. Jejich seznam je součástí tohoto dokumentu.

## 2 Licence

Knihovna je určena pouze pro firmu Personna International CZ s.r.o. pro využití na všech projektech, které slouží k interním účelům firmy Personna International CZ s.r.o.

Použití knihovny podléhá v dalších bodech platným všeobecným obchodním podmínkám firmy Mepatek s.r.o. uvedeným na [www.mepatek.cz](http://www.mepatek.cz) a platné legislativě na území ČR.

## 3 Instalace

### 3.1 Composer

Nejjednodušší způsob instalace je pomocí programu **composer** (dokumentace <https://getcomposer.org/doc/>, stažení <https://getcomposer.org/download/>).

Instalace se provede spuštěním příkazu

```
composer require mepatek/totelib
```

Do složky vendor se nainstaluje knihovna ToteLib včetně všech dalších potřebných souvisejících knihoven a souborů.

Aktualizace knihovny se provede spuštěním příkazu

```
composer update mepatek/totelib
```

### 3.2 Stažení z GitHub

Druhou možností (nedoporučujeme) je stáhnout kompletní sadu souborů z GitHub z poslední release. Aktuální release je k dispozici zde: [https://github.com/Mepatek/totelib/releases/download/v1.0/ToteLibFull\\_v1.0.zip](https://github.com/Mepatek/totelib/releases/download/v1.0/ToteLibFull_v1.0.zip). Soubor obsahuje zazipovanou strukturu složek, které je nutné nahrát do kořenové složky projektu.

## 4 Použití

Knihovna obsahuje třídu Mepatek\ToteLib\ToteLibFacade. Knihovna využívá pro svůj chod tyto knihovny:

NÁZEV KNIHOVNY	ZDROJOVÉ SOUBORY	DOKUMENTACE
NETTE/UTILS	<a href="https://github.com/nette/utils">https://github.com/nette/utils</a>	<a href="https://doc.nette.org/cs/3.0/utils">https://doc.nette.org/cs/3.0/utils</a>
VLUCAS/PHPDOTENV	<a href="https://github.com/vlucas/phpdotenv">https://github.com/vlucas/phpdotenv</a>	<a href="https://github.com/vlucas/phpdotenv/blob/master/README.md">https://github.com/vlucas/phpdotenv/blob/master/README.md</a>
DIBI/DIBI	<a href="https://github.com/dg/dibi">https://github.com/dg/dibi</a>	<a href="https://dibiphp.com/cs/documentation">https://dibiphp.com/cs/documentation</a>

K pohodlnému přístupu k databázi doporučujeme využívat funkci Dibi knihovny.

Knihovna ToteLib vrací objekt Dibi connection, se kterým lze dále pracovat velmi jednoduše a efektivně dle dokumentace na <https://dibiphp.com/cs/documentation>.

Knihovna také vrací PDO objekt, který nedoporučujeme používat. Dokumentace jak s daty pomocí PDO pracovat je zde: <https://www.php.net/manual/en/ref.pdo-sqlsrv.php>

## 4.1 Konfigurace

Konfigurace připojení se realizuje pomocí .env souboru. Jeho umístění je libovolné, cestu k němu je nutné předat při vytváření objektu (new ToteLibFacade(\$dir)).

V souboru jsou tyto konfigurační proměnné:

```
TOTE_SERVER=
TOTE_USERNAME=
TOTE_PASSWORD=
TOTE_DATABASE=
```

Proměnná TOTE\_DATABASE je nepovinná, slouží na určení do jaké databáze se má skript přepnout (například pokud by existovala nějaká testovací verze TOTE databáze, její jméno se lze zapsat).

Více informací o .env souboru zde: <https://github.com/vlucas/phpdotenv/blob/master/README.md>

## 4.2 Základní použití knihovny

Třída vyžaduje při inicializaci adresář, kde je umístěn .env soubor, aby byla schopna načíst konfiguraci. Dobrým zvykem je mít .env soubor v kořenu projektu.

Pokud je složka vendor také v kořenu projektu, inicializuje se knihovna ve skriptu v kořenu projekt takto:

```
// replace by real path to vendor/autoload.php
require_once __DIR__ . "/vendor/autoload.php";

// initialize library, load environment variable from __DIR__ (replace by real path to
// .env file)
$toteLib = new \Mepatek\ToteLib\ToteLibFacade(__DIR__);
```

Pokud je složka vendor jinde, změní se cesta u require\_once.

Pokud je soubor .env jinde, změní se cesta v konstruktoru třídy ToteLibFacade( jina\_cesta ).

Základní použití knihovny s využitím Dibi je:

```
// get Dibi\Result from TOTE database WHERE name start with 'likeNameString'
$result = $toteLib->query("SELECT id, name FROM TableName WHERE name LIKE %like~",
"likeNameString");
// get all rows from result
$result = $result->fetchAll();
// or iterate row by row
foreach ($result as $row) {
    echo $row->id;
    echo $row->name;
}
```

\$result = \$toteLib->query(\$sql, \$param) vrátí objekt Result, kterým lze iterovat nebo obsahuje mimo jiné metody fetchAll, fetchPairs, fetchSingle. Vše je podrobně popsáno v dokumentaci <https://dibiphp.com/cs/documentation>

Pro zadávání podmínek a parametrů do dotazu doporučujeme (kvůli bezpečnosti, zejména problému s SQL injection ([https://cs.wikipedia.org/wiki/SQL\\_injection](https://cs.wikipedia.org/wiki/SQL_injection)) pomocí modifikátorů.

Základní modifikátory jsou:

%s	string
%sN	string, ale "" se přeloží jako NULL
%bin	binární data
%b	boolean
%i	integer
%iN	integer, ale 0 se přeloží jako NULL
%f	float
%d	datum (očekává DateTime, string nebo UNIX timestamp)
%dt	datum & čas (očekává DateTime, string nebo UNIX timestamp)
%n	identifikátor, tedy název tabulky či sloupce
%SQL	SQL – přímo vloží do SQL (alternativou je Dibi\Literal)
%ex	expanduje pole
%lmt	speciální – doplní do dotazu LIMIT
%ofs	speciální – doplní do dotazu OFFSET

Další modifikátory a příklady použití jsou zde: <https://dibiphp.com/cs/documentation>

Příklady základní práce s knihovnou je demonstrována v souboru, který je součástí zdrojů knihovny `examples/ToteLibBasics.php`

## 4.3 Rozšířené možnosti knihovny

Rozšířené možnosti knihovny jsou:

```
// get Dibi\Database object with
$dbaseInfo = $toteLib->getDatabaseInfo();
```

Vrací objekt `Dibi\Database`, který obsahuje základní informace o připojené databázi.

Popis metod zde: <https://api.dibiphp.com/4.0/Dibi/Reflection/Database.html>

```
// get array of Dibi\Table object
$tables = $toteLib->getTables();
```

Vrací pole s objekty `Dibi\Table`, které obsahují informace o tabulkách (název tabulky, seznam sloupců, seznam indexů, primární klíč, seznam klíčů, zda jde o view).

Popis metod zde: <https://api.dibiphp.com/4.0/Dibi/Reflection/Table.html>

```
// get array of table names (string)
$tableNames = $toteLib->getTableNames();
```

Vrací pole s názvy (řetězec) tabulek v připojené databázi.

```
// get Dibi\Result from TOTE database WHERE name start with 'likeNameString'
$result = $toteLib->query("SELECT id, name FROM TableName WHERE name LIKE %like~",
"likeNameString");
```

Vrací `Dibi\Result`, který obsahuje všechny hodnoty id a name vyhovující podmínce, že jméno začíná na `likeNameString`.

Popis metod třídy `Result` zde: <https://api.dibiphp.com/4.0/Dibi/Result.html>

```
// get all rows from result
$result = $result->fetchAll();
```

Vrací všechny řádky z resultu jako pole. Každý řádek v dotazu je řádek v poli.

## 4.4 Pomocné funkce

Pro ladění jsou v knihovně tři funkce, které pomohou s výpisem dat z tabulek.

```
// get Dibi result with all data from 'table'
$result = $toteLib->getTableResult("table");
```

Vrátí Dibi result se všemi daty z tabulky table.

```
// get Nette Html element table with all data from 'table'
$htmlTable = $toteLib->getFullTableAsHtmlTable("table");
// and show it
echo $htmlTable;
```

Vrátí Nette Html element table obsahující všechna data z tabulky table. Vypsát data lze použitím příkazu echo.

Dokumentace k Html třídě zde: <https://doc.nette.org/cs/3.0/html-elements>

```
// get Nette Html element table with id and name data from 'table' where date =
current date
$htmlTable = $toteLib->getQueryResultAsHtmlTable('SELECT id, name FROM table WHERE
date = %d', new DateTime());
// and show it
echo $htmlTable;
```

Vrátí Nette Html element table obsahující id a name data z tabulky table, které mají hodnotu date rovnou dnešnímu dni. Vypsát data lze použitím příkazu echo.

Dokumentace k Html třídě zde: <https://doc.nette.org/cs/3.0/html-elements>

Příklady použití pomocných funkcí je demonstrována v souboru, který je součástí zdrojů knihovny examples/ToteLibHelpers.php

## 4.5 PDO

PDO objekt nedoporučujeme používat, kvůli bezpečnosti a složitějšímu volání funkcí.

```
// get PDO object
$pdo = $toteLib->getPdo();
```

Vrátí objekt PDO spojený do TOTE databáze.

```
// run query
$pdo->query("SELECT * FROM table");
```

Provede PDO dotaz.

Dokumentace k PDO zde: <https://www.php.net/manual/en/ref.pdo-sqlsrv.php>

Příklady práce PDO s knihovnou je demonstrována v souboru, který je součástí zdrojů knihovny examples/ToteLibPdo.php

## 4.6 Adminer

Pro práci s databází, testování SQL dotazů je možné vytvořit php soubor, který zobrazí nástroj Adminer od Jakuba Vrány. Více informací zde: <https://www.adminer.org/cs/>

PHP skript by měl obsahovat toto:

```
<?php
declare(strict_types=1);

// replace by real path to vendor/autoload.php
require_once __DIR__ . "/vendor/autoload.php";

// create empty adminer.css if not exist
touch(__DIR__ . '/adminer.css');
// initialize library, load environment variable from __DIR__ (replace by real path to
.env file)
$toteLib = new \Mepatek\ToteLib\ToteLibFacade(__DIR__);

//redirect to adminer
require_once __DIR__ . "/vendor/mepatek/totelib/src/ToteAdminer.php";
exit();
```

Pokud se jedná například o soubor ToteAdminer.php v kořenu projektu, pak jen stačí z prohlížeče <http://serverSProjektem/ToteAdminer.php> a zobrazí se přihlašovací stránka, která má předvyplněné přihlašovací údaje z konfigurace v .env souboru.

Příklad volání Adminera je demonstrována v souboru, který je součástí zdrojů knihovny examples/ToteLibAdminer.php

## 5 Dokumentace všech funkcí

Vytvoření objektu Mepatek\ToteLib\ToteLibFacade:

```
$toteLib = new \Mepatek\ToteLib\ToteLibFacade($dir);
```

\$dir je cesta k .env souboru

```
/**
 * @return PDO
 * @see https://www.php.net/manual/en/ref.pdo-sqlsrv.php
 */
public function getPdo(): PDO
```

Vrací PDO objekt. <https://www.php.net/manual/en/ref.pdo-sqlsrv.php>

```
/**
 * Get Dibi Connection object
 *
 * @return Connection
 * @throws Exception
 * @see https://dibiphp.com/cs/documentation
 */
public function getDibi(): Connection
```

Vrací Dibi Connection objekt. <https://api.dibiphp.com/4.0/Dibi/Connection.html>

```
/**
 * Get Dibi driver
 *
```

```
* @return Driver
* @throws Exception
*/
public function getDriver(): Driver
```

Vrací Dibi Driver objekt. <https://api.dibiphp.com/4.0/Dibi/Driver.html>

```
/**
 * Get Dibi result from $sql and parameters
 *
 * @param string $sql
 * @param mixed $parameter_1
 * @param mixed $parameter_2
 * @param mixed $parameter_3
 * @param mixed $parameter_n
 * @return Result
 * @throws Exception
 */
```

```
public function query(...$args): Result
```

Provede query pomocí Dibi Connection. Funkce očekává jako první argument SQL dotaz s parametry viz dokumentace <https://dibiphp.com/cs/documentation> zejména část Modifikátory a Modifikátory polí. Další argumenty jsou parametry SQL dotazu, pořadí odpovídá v SQL dotazu.

Funkce vrací Dibi Result objekt. <https://api.dibiphp.com/4.0/Dibi/Result.html>

```
/**
 * Get Dibi Database reflection object.
 *
 * @return Database
 * @throws Exception
 */
```

```
public function getDatabaseInfo(): Database
```

Vrací Dibi Database reflection objekt. <https://api.dibiphp.com/4.0/Dibi/Reflection/Database.html>

```
/**
 * Get all Dibi Tables in connected database
 *
 * @return Table[]
 * @throws Exception
 */
```

```
public function getTables(): array
```

Vrací pole s Dibi Table objekty. <https://api.dibiphp.com/4.0/Dibi/Reflection/Table.html>

```
/**
 * Get all table names in connected database
 *
 * @return string[]
 * @throws Exception
 */
```

```
public function getTableNames(): array
```

Vrací pole s názvy tabulek.

```
/**
 * Get Dibi Result object with all data from table $table
 *
 * @param string $table
 * @return Result
 * @throws Exception
 */
```



```
*/  
public function getTableResult(string $table): Result
```

Vrací objekt Dibi Result se všemi daty z tabulky \$table.

```
/**  
 * Get table Html with all data from table  
 *  
 * @param string $table  
 * @return Html  
 * @throws Exception  
 * @see https://doc.nette.org/cs/3.0/html-elements  
 */
```

```
public function getFullTableAsHtmlTable(string $table): Html
```

Vrací Nette Html element objekt obsahující tabulku se všemi daty tabulky \$table. <https://doc.nette.org/cs/3.0/html-elements>

```
/**  
 * Get table Html with all data from query result  
 *  
 * @param string $sql  
 * @param mixed $parameter_1  
 * @param mixed $parameter_2  
 * @param mixed $parameter_3  
 * @param mixed $parameter_n  
 * @return Html  
 * @throws Exception  
 * @see https://doc.nette.org/cs/3.0/html-elements  
 */
```

```
public function getQueryResultAsHtmlTable(...$arg): Html
```

Vrací Nette Html element objekt obsahující tabulku se všemi daty z dotazu. <https://doc.nette.org/cs/3.0/html-elements>

Funkce očekává jako první argument SQL dotaz s parametry viz dokumentace <https://dibiphp.com/cs/documentation> zejména část Modifikátory a Modifikátory polí. Další argumenty jsou parametry SQL dotazu, pořadí odpovídá v SQL dotazu.