Constructor

Methods

Usage

Jig

The Jig class provides a simple way to store arrays of data into flat ASCII files.

Namespace: `\DB`
File location: `lib/db/jig.php`

Constructor

```
$db = new \DB\Jig ( string $dir [, int $format = \DB\Jig::FORMAT_JSON ] );
```

`$dir` defines the storage directory name. It can be relative to the base path or absolute.

`$format` defines the storage format. Possible values are:

- `\DB\Jig::FORMAT_JSON` which stores data in JSON format.
- `\DB\Jig::FORMAT_Serialized` which serializes data. Depending on the value of the SERIALIZER (quick-reference#serializer) system variable, data is stored using the standard PHP serialization format, the igbinary format or JSON.

N.B.1: Like every F3 directory name, `$dir` must end with a trailing slash!
N.B.2: The directory is automatically created if not already existing.

Methods

read

Read data from a Jig database file

```
array read ( string $file )
```

This method retrieves the data stored in a given Jig database file.

For example:

```
$data = $db->read('team');
echo $data['Robert']['birth']; // 82 (see below)
```

write

Write data to a Jig database file

```
int|FALSE write ( string $file, array $data )
```

This method writes data to a Jig database file and returns the number of bytes that were written to the file, or `FALSE` on failure.

For example:

```
$db=new \DB\Jig('jig/',\DB\Jig::FORMAT_Serialized);
$team=array(
  'Jimmy'   => array('birth'=>86,'instr'=>'guitars'),
  'Robert'  => array('birth'=>82,'instr'=>'vocals'),
  'John'    => array('birth'=>88,'instr'=>'drums'),
  'Anna'    => array(            'instr'=>'keyboards'),
);
echo $db->write('team',$team); // outputs 231 (int)
```

drop

Clean storage

```
int drop ( )
```

Erases all the data stored in the storage *directory*.

Example:

```
$db=new \DB\Jig('jig/'); // jig/ directory is empty
$db->write('file1.dat',$data1); // jig/ contains file1.dat
$db->write('file2.dat',$data2); // jig/ contains file1.dat and file2.dat
$db->drop(); // jig/ directory is now empty
```

dir

Return database storage directory

```
string dir ( )
```

Actually, simply returns the `$dir` value that was specified to the constructor (jig#constructor).

jot

Jot down a log entry

```
null jot ( string $frame )
```

Add the given `$frame` to the inner log. Entries are prefixed with a RFC 2822 formatted date.

log

Return Jig profiler results

```
string log ( )
```

Example:

```
$db->jot('Merry Christmas');
echo $db->log();  // "Wed, 25 Dec 2013 08:27:58 +0100 Merry Christmas"
```

uuid

Return UUID (Universally Unique IDentifier) connection hash

```
string uuid ( )
```

Example:

```
echo $db->uuid();  // e.g. "0dso6nqcdhr"  (string, length 11)
```

Usage

The Jig DB is not meant to be used directly. Instead have a look at our powerful Jig Mapper (jig-mapper). To initalize the database, simply add some records with the mapper, i.e.:

```
$mapper = new \DB\Jig\Mapper($db, $file);
$mapper->username = 'userA';
$mapper->password = '57d82jg05';
$mapper->save();
$mapper->reset();
$mapper->username = 'userB';
$mapper->password = 'kbjd94973';
$mapper->save();
```

In case you used the default JSON style Jig DB, the resulting json db file structure looks like this:

```
{
    "548723b9f06c78.10153217": {
        "username": "userA",
        "password": "57d82jg05"
    },
    "54f9c763934745.48648465": {
        "username": "userB",
        "password": "kbjd94973"
    }
}
```