Database 13

Database Functionality

> This only applies to Framework 13.0.121 or higher

> If you haven't already done so please brush up on Database in FreePBX 12 which also applies to this article

Starting with FreePBX 13 the development team has completely removed the need for PearDB. PearDB was a Database abstraction layer that FreePBX used since it's inception. Over the years the project was depreciated in favor of MDB2, however at the same time the PHP developers added a new abstraction layer called PDO. FreePBX 13 now uses the abstraction layer called PDO with a wrapper for old PearDB calls.

> Our complete PearDB to PDO wrapper can be viewed here: http://git.freepbx.org/projects/FREEPBX/repos/framework/browse/amp_conf/htdocs/admin/libraries/DB.class.php
>
> It's 100% open source. Feel free to use it in whatever project you need

Creating/Altering/Updating a Database Programmatically

Starting in Framework 13.0.121 you can now let FreePBX automatically/automagically maintain your Database table structure. Simply call the "migrate" method of FreePBX's database abstraction engine and pass the parameter the name of your table:

```
$table = $this->FreePBX->Database->migrate("meetme");
```

You will then be passed back a "Database\Migrate" object to work with.

Adding/Modifying Columns

Columns are updated through a multi dimensional array mechanism, where the "key" is the column name and the values are an array of options

```
$cols = array(
        "exten" => array(
                "type" => "string",
                "length" => 50
        )
);
```

The list of options is as follows:

- **type** (string): The column type, can be:
  - **string:** Type that maps to a SQL VARCHAR
  - **integer:** Type that maps to a SQL INT

    - **smallint:** Type that maps to a database SMALLINT
    - **bigint:** Type that maps to a database BIGINT
    - **boolean:** Type that maps to a SQL boolean or equivalent (TINYINT)
    - **decimal:** Type that maps to a SQL DECIMAL
    - **text:** Type that maps to a SQL CLOB
    - **float:** Type that maps to a SQL Float (Double Precision)
    - **blob:** Type that maps to a SQL BLOB
    - **date:** Type that maps to a SQL DATETIME
    - **time:** Type that maps to a SQL TIME
    - **datetime:** Type that maps to a SQL DATETIME/TIMESTAMP
- **primaryKey** (boolean): When set to true this column becomes the primary key
- **notnull** (boolean): Whether the column is nullable or not. Defaults to `true`.
- **default** (integer|string): The default value of the column if no value was specified. Defaults to `null`.
- **autoincrement** (boolean): Whether this column should use an autoincremented value if no value was specified. Only applies to Doctrine's `smallint`, `integer` and `bigint` types. Defaults to `false`.
- **length** (integer): The maximum length of the column. Only applies to Doctrine's `string` and `binary` types. Defaults to `null` and is evaluated to `255` in the platform.
- **fixed** (boolean): Whether a `string` or `binary` Doctrine type column has a fixed length. Defaults to `false`.
- **precision** (integer): The precision of a Doctrine `decimal` or `float` type column that determines the overall maximum number of digits to be stored (including scale). Defaults to `10`.
- **scale** (integer): The exact number of decimal digits to be stored in a Doctrine `decimal` or `float` type column. Defaults to `0`.
- **customSchemaOptions** (array): Additional options for the column that are supported by all vendors:
- **unique** (boolean): Whether to automatically add a unique constraint for the column. Defaults to `false`.

Adding/Modifying Indexes

Indexes (Besides the Primary Key) are updated through a multi dimensional array mechanism, where the "key" is the index name and the values are an array of options

```
$indexes = array(
        "index1" => array(
                "type" => "unique",
                "cols" => array(
                        "music",
                        "users",
                        "language"
                )
        )
);
```

The list of options is as follows:

- **type** (string): The index type, can be:
    - **unique:** an index where all rows of the index must be unique
    - **index:** a normal non-unique index. Non-distinct values for the index are allowed, so the index may contain rows with identical values in all columns of the index. These indexes don't enforce any restraints on your data so they are used only for making sure certain queries can run quickly.
    - **fulltext:** only useful for full text searches done with the MATCH() / AGAINST() clause
- **cols** (array): The columns this key will affect

As of framework version 13.0.190.19 (Thanks to miken32) you can also add foreign keys using the indexes array:

The list of options is as follows:

- **type** (string): The index type, can be:
    - **unique:** an index where all rows of the index must be unique
    - **index:** a normal non-unique index. Non-distinct values for the index are allowed, so the index may contain rows with identical values in all columns of the index. These indexes don't enforce any restraints on your data so they are used only for making sure certain queries can run quickly.
    - **fulltext:**only useful for full text searches done with the `MATCH()` / `AGAINST()` clause
    - **foreign:** a foreign key index
- **cols** (array): The columns this key will affect
- **foreigntable** (string): The foreign table this key will link to
- **foreigncols** (array): The foreign table columns the key will link to

- **options** (array): For foreign key indices, an array of strings with keys `onDelete` or `onUpdate`. Each can have a values of `RESTRICT`, `CASCADE`, `SET NULL`, `NO ACTION`, or `SET DEFAULT`

Apply

To apply your two arrays you simply run the modify method against the table object

```
$table->modify($cols,$indexes);
```

Or if you have no indexes:

```
$table->modify($cols);
```

Make sure to unset when you are done to free up memory

```
unset($table);
```

Complete

When finished you will have something that looks like this (Note this is NOT how the meetme table is constructed. It's for demo purposes only):

```
$table = $this->FreePBX->Database->migrate("meetme");
$cols = array(
        "exten" => array(
                "type" => "string",
                "length" => 50,
                "primaryKey" => true
        ),
        "options" => array(
                "type" => "string",
                "length" => 15,
                "notnull" => false,
        ),
        "userpin" => array(
                "type" => "string",
                "length" => 50,
                "notnull" => false,
        ),
        "adminpin" => array(
                "type" => "string",
                "length" => 50,
                "notnull" => false,
        ),
        "description" => array(
                "type" => "string",
                "length" => 50,
                "notnull" => false,
        ),
        "joinmsg_id" => array(
                "type" => "integer",
                "notnull" => false,
        ),
        "music" => array(
                "type" => "string",
                "length" => 80,
                "notnull" => false,
        ),
        "users" => array(
                "type" => "smallint",
                "unsigned" => false,
                "default" => 0,
                "notnull" => false
        ),
        "language" => array(
                "type" => "string",
                "length" => 10,
                "default" => "",
        ),
);
$table->modify($cols);
unset($table);
```

Which looks like this in MySQL

```
mysql> describe meetme;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| exten       | varchar(50) | NO   | PRI | NULL    |       |
| options     | varchar(15) | YES  |     | NULL    |       |
| userpin     | varchar(50) | YES  |     | NULL    |       |
| adminpin    | varchar(50) | YES  |     | NULL    |       |
| description | varchar(50) | YES  |     | NULL    |       |
| joinmsg_id  | int(11)     | YES  |     | NULL    |       |
| music       | varchar(80) | YES  |     | NULL    |       |
| users       | smallint(6) | YES  |     | 0       |       |
| language    | varchar(10) | NO   |     |         |       |
+-------------+-------------+------+-----+---------+-------+
9 rows in set (0.00 sec)
```

Automatically generate PHP table creation code

You can also easily generate the PHP code needed to create tables in your modules. First create the table in your database (MySQL/MariaDB preferred). Then run:

```
fwconsole doctrine <tablename>
```

An example of the output is pasted below. You can copy this code and apply it directly to your install() method or install.php

```
[root@freepbxdev4 ~]# fwconsole doctrine outbound_routes
$table = \FreePBX::Database()->migrate("outbound_routes");
$cols = array (
  'route_id' =>
  array (
    'type' => 'integer',
    'primaryKey' => true,
    'autoincrement' => true,
  ),
  'name' =>
  array (
    'type' => 'string',
    'length' => '40',
    'notnull' => false,
  ),
  'outcid' =>
  array (
    'type' => 'string',
    'length' => '40',
    'notnull' => false,
  ),
  'outcid_mode' =>
  array (
```

```
    'type' => 'string',
    'length' => '20',
    'notnull' => false,
  ),
  'password' =>
  array (
    'type' => 'string',
    'length' => '30',
    'notnull' => false,
  ),
  'emergency_route' =>
  array (
    'type' => 'string',
    'length' => '4',
    'notnull' => false,
  ),
  'intracompany_route' =>
  array (
    'type' => 'string',
    'length' => '4',
    'notnull' => false,
  ),
  'mohclass' =>
  array (
    'type' => 'string',
    'length' => '80',
    'notnull' => false,
  ),
  'time_group_id' =>
  array (
    'type' => 'integer',
    'notnull' => false,
  ),
  'dest' =>
  array (
    'type' => 'string',
    'length' => '255',
    'notnull' => false,
  ),
  'time_mode' =>
  array (
    'type' => 'string',
    'length' => '20',
    'notnull' => false,
    'default' => 'time-group',
  ),
  'calendar_id' =>
  array (
    'type' => 'integer',
    'notnull' => false,
  ),
  'calendar_group_id' =>
  array (
```

```
    'type' => 'integer',
    'notnull' => false,
  ),
);

$indexes = array (
);
$table->modify($cols, $indexes);
unset($table);
```

Creating/Altering/Updating a Database Through module.xml

Starting in Framework 13 we've implemented an even easier way to generate database tables. You can now generate database tables for your module through the module.xml! First read through Database 13#Creating/Altering/Updating a Database Programmatically to understand how simple database creation works. For more information about the structure of module.xml please read through: module.xml

Automatically generate Module.xml table creation XML

You can also easily generate the XML needed to create tables in your modules. First create the table in your database (MySQL/MariaDB preferred). Then run:

```
fwconsole doctrine <tablename> --format=xml
```

An example of the output is pasted below. You can copy this code and apply it directly to module.xml

```
[root@freepbxdev4 ~]# fwconsole doctrine outbound_routes --format=xml
<database>
  <table name="outbound_routes">
    <field name="route_id" type="integer" primaryKey="true" autoincrement="
true"/>
    <field name="name" type="string" length="40" notnull="false"/>
    <field name="outcid" type="string" length="40" notnull="false"/>
    <field name="outcid_mode" type="string" length="20" notnull="false"/>
    <field name="password" type="string" length="30" notnull="false"/>
    <field name="emergency_route" type="string" length="4" notnull="false"
/>
    <field name="intracompany_route" type="string" length="4" notnull="
false"/>
    <field name="mohclass" type="string" length="80" notnull="false"/>
    <field name="time_group_id" type="integer" notnull="false"/>
    <field name="dest" type="string" length="255" notnull="false"/>
    <field name="time_mode" type="string" length="20" default="time-group"
notnull="false"/>
    <field name="calendar_id" type="integer" notnull="false"/>
    <field name="calendar_group_id" type="integer" notnull="false"/>
  </table>
</database>
```