REDBEAN 4 UPGRADE GUIDE

This describes how to upgrade from RedBean 3.5 to RedBean 4.3.4.

Installing Redbean 4

Make sure to remove / stop including the old v3 library.

Then decide if you want to include RedBean 4 manually or via Composer.

Via Composer

- Add the new version via Composer:
  - `composer require gabordemooij/redbean ^4`
- Add `use \RedBeanPHP\R as R;` at the top of your app.
  - Or, alternatively, `class_alias('\RedBeanPHP\R','\R');`
  - This will avoid `Uncaught Error: Class 'R' not found` errors.

Note about namespaces: RedBean's default namespace is

- `\RedBeanPHP\R` when installed via Composer;
- `\R` when installed manually (rb.php).

If you're including RedBean via Composer and want to use the short form shown throughout the docs, then you need to use `use` or `class_alias` as shown above. Otherwise, the alternative is to use the fully qualified class wherever you use RedBean.

(If including RedBean manually, do NOT use this `use` / `alias`.)

Manually

- Simply include `rb.php`.

Preloader

If you were using the Preloader in RedBean 3.5, then install the preloader extension for RedBean 4, available from https://github.com/gabordemooij/RB4Plugins

Note: if it's missing but you use `R::preload` anyway, then you'll get a misleading error message: `Plugin 'preload' does not exist, add this plugin using: R::ext('preload')`. Adding `R::ext('preload')` won't help (indeed, the example that comes with Preloader.php does not use it); adding the plugin will.

Configuring RedBean

Decide if you want to define your RedBean models with or without a namespace, and choose the prefix used for your models (both are connected.) Some examples:

Default prefix / No namespace:

```
define('REDBEAN_MODEL_PREFIX', '\\Model_');
    /**
     * The default. Then:
     * - Don't use a namespace in the model class!
     * - Declare the model class like so:
     *   - `class Model_Book extends \RedBeanPHP\SimpleModel`
     * - (If including the library manually, you could also simply use:
     *   - `class Model_Book extends RedBean_SimpleModel`
     *   - But I don't recommend it. It will only cause confusion if you later
     *     switch to using composer.)
```

With namespace and relevant prefix:

```
define('REDBEAN_MODEL_PREFIX', '\\Model\\');
    /**
     * This is if you'd like all your models to be in a `Model` namespace.
     * Then make sure to:
     * - Set the namemespace in the class:
     *   - `namespace Model;`
     * - Declare the class properly:
     *   - `class Book extends \RedBeanPHP\SimpleModel`
     */
```

Note: the readme mentions `class \Model\Band extends \RedBeanPHP\SimpleModel` which is inaccurate and will cause "Parse error: syntax error, unexpected '\' (T*NS*SEPARATOR), expecting identifier (T_STRING)".

Without any prefix:

```
define('REDBEAN_MODEL_PREFIX', '');
    /**
     * If you'd like your model class to be just `Book`:
     * `class Book extends RedBean_SimpleModel`.
     */
```

New to namespaces?

If you're now using a namespace for your models while in the past it did not, then you also need to either:

- Declare the classes you're using at the top of your file with `use`. e.g.:
  - `use \Datetime;`
  - `use \R;`
- Or, use the full class qualifier when referring to external classes inside your model.

Other things worth knowing

SQLHelper

The SQLHelper (that was accessed via `R::$f` in RedBean 3) is gone. Either:

- Rewrite those into queries using `R::getAll()`;

- Use the *SQLHelper* RB4 plugin: https://github.com/gabordemooij/RB4Plugins

Namespace and models

If getting confused about namespaces and as to whether or not your model is being loaded, `class_exists` will come in handy, e.g.:

- `var_dump(class_exists('Model_Book'));`
- `var_dump(class_exists('Model\Book'));`
- `var_dump(class_exists('Book'));`

Debugging the FUSE model

In development, I recommend turning on error messages to know if a FUSE model method does not exist. Easy to do with:

```
R::setErrorHandlingFUSE(RedBeanPHP\OODBBean::C_ERR_EXCEPTION);
```

(See the API for the other possible error levels.)

Note: `setErrorHandlingFUSE` may throw misleading exceptions on foreign keys for lack of preload. In other words: if RedBean failed to preload foreign key `foo`, it may throw:
`An exception has been thrown during the rendering of a template ("FUSE: method does not exist in model: foo")`
… As if `foo` was a method, even though it's actually a foreign key. (Had this occur when trying to load a foreign key from within a Twig template without preloading; with preloading, no issue.)

See also

- CHANGELOG
- README
- setErrorHandlingFUSE in the API
- Models
- Legacy downloads for RedBean 4
- RB4Plugins