NAME

Padre::DB::Plugin − Padre::DB class for the `plugin` table

SYNOPSIS

TO BE COMPLETED

DESCRIPTION

TO BE COMPLETED

METHODS

select

```
# Get all objects in list context
my @list = Padre::DB::Plugin->select;

# Get a subset of objects in scalar context
my $array_ref = Padre::DB::Plugin->select(
    'where name > ? order by name',
    1000,
);
```

The `select` method executes a typical SQL `SELECT` query on the `plugin` table.

It takes an optional argument of a SQL phrase to be added after the `FROM plugin` section of the query, followed by variables to be bound to the placeholders in the SQL phrase. Any SQL that is compatible with SQLite can be used in the parameter.

Returns a list of `Padre::DB::Plugin` objects when called in list context, or a reference to an ARRAY of `Padre::DB::Plugin` objects when called in scalar context.

Throws an exception on error, typically directly from the DBI layer.

count

```
# How many objects are in the table
my $rows = Padre::DB::Plugin->count;

# How many objects
my $small = Padre::DB::Plugin->count(
    'where name > ?',
    1000,
);
```

The `count` method executes a `SELECT COUNT(*)` query on the `plugin` table.

It takes an optional argument of a SQL phrase to be added after the `FROM plugin` section of the query, followed by variables to be bound to the placeholders in the SQL phrase. Any SQL that is compatible with SQLite can be used in the parameter.

Returns the number of objects that match the condition.

Throws an exception on error, typically directly from the DBI layer.

new

TO BE COMPLETED

The `new` constructor is used to create a new abstract object that is not (yet) written to the database.

Returns a new `Padre::DB::Plugin` object.

create

```
my $object = Padre::DB::Plugin->create(
    name    => 'value',
    version => 'value',
    enabled => 'value',
    config  => 'value',
);
```

The `create` constructor is a one-step combination of `new` and `insert` that takes the column parameters, creates a new `Padre::DB::Plugin` object, inserts the appropriate row into the `plugin` table, and then returns the object.

If the primary key column `name` is not provided to the constructor (or it is false) the object returned will have `name` set to the new unique identifier.

Returns a new `plugin` object, or throws an exception on error, typically from the DBI layer.

insert

```
$object->insert;
```

The `insert` method commits a new object (created with the `new` method) into the database.

If a the primary key column `name` is not provided to the constructor (or it is false) the object returned will have `name` set to the new unique identifier.

Returns the object itself as a convenience, or throws an exception on error, typically from the DBI layer.

delete

```
# Delete a single instantiated object
$object->delete;

# Delete multiple rows from the plugin table
Padre::DB::Plugin->delete('where name > ?', 1000);
```

The `delete` method can be used in a class form and an instance form.

When used on an existing `Padre::DB::Plugin` instance, the `delete` method removes that specific instance from the `plugin`, leaving the object intact for you to deal with post-delete actions as you wish.

When used as a class method, it takes a compulsory argument of a SQL phrase to be added after the `DELETE FROM plugin` section of the query, followed by variables to be bound to the placeholders in the SQL phrase. Any SQL that is compatible with SQLite can be used in the parameter.

Returns true on success or throws an exception on error, or if you attempt to call delete without a SQL condition phrase.

truncate

```
# Delete all records in the plugin table
Padre::DB::Plugin->truncate;
```

To prevent the common and extremely dangerous error case where deletion is called accidentally without providing a condition, the use of the `delete` method without a specific condition is forbidden.

Instead, the distinct method `truncate` is provided to delete all records in a table with specific intent.

Returns true, or throws an exception on error.

INTERFACE

Accessors

name

```
if ( $object->name ) {
    print "Object has been inserted\n";
} else {
    print "Object has not been inserted\n";
}
```

Returns true, or throws an exception on error.

```
REMAINING ACCESSORS TO BE COMPLETED
```

SQL

The `plugin` table was originally created with the following SQL command.

```
CREATE TABLE plugin (
      name VARCHAR(255) PRIMARY KEY,
      version VARCHAR(255),
      enabled BOOLEAN,
      config TEXT
)
```

SUPPORT

`Padre::DB::Plugin` is part of the Padre::DB API.

See the documentation for Padre::DB for more information.

AUTHOR

Adam Kennedy

COPYRIGHT