JTable

Overview

The JTable class is an implementation of the Active Record design pattern. It is used throughout Joomla! for creating, reading, updating, and deleting records in the database table.

When properly extended, JTable gives you all of the basic functions you need for managing and retrieving records in a database table. Member functions take care of the rest when you add member variables, the table name, and the key column.

Writing an extension of JTable

To use JTable, create an extension of the class. In this example, we have a database table containing recipes.

```
<?php

defined('_JEXEC') or die();

class TableRecipes extends JTable
{
 var $id = null;
 var $ingredients = null;
 var $instructions = null;
 var $serves = null;
 var $difficulty = null;
 var $prep_time = null;
 var $cook_time = null;
 var $published = 0;

 function __construct(&$db)
 {
  parent::__construct( '#__recipes', 'id', $db );
 }
}
```

When naming your class extension, the convention is to prefix it with 'Table', then follow with a CamelCased version of the table's name. All of the member variables of your class should match the column names in the database. The default values should be valid according to the table schema. For instance, if you have columns that are NOT NULL, you must use a value

other than 'null' as the default.

Finally, create a constructor for the class that accepts a reference to the current database instance. This will call the parent constructor which needs the name of the table, the name of the primary key column, and the database instance. The name of the table uses #__ instead of jos_, as the administrator can pick any table prefix desired during Joomla! installation.

If you were using this class as a part of a component called 'Recipes', you would place this code in the file /administrator/components/com_recipes/tables/recipes.php.

Using a JTable class extension

Once the table class is in place, you can use it in any Joomla! extension. To include the file, place this line in your extension's source code (use com_nameofyourcomponent in place of com_recipes):

```
JTable::addIncludePath(JPATH_ADMINISTRATOR.DS.'components'.DS.'com_recipes'.DS.'tables');
```

To get an instance of the object, use this code:

```
$row =& JTable::getInstance('recipes', 'Table');
```

Notice that the lowercase version of the suffix of your class name is used as the first parameter, with the prefix 'Table' as the second. Also, the getInstance() member function of JTable returns the object by reference instead of value.

In a model class (extends JModel) you can also use:

```
$row =& $this->getTable('recipes');
```

Notice that if you have not used the standard naming convention, you can supply the class prefix as the optional second parameter.

Create/Update

In a typical situation, you will have an HTML form submitted by the user which PHP will interpret for you as an associative array. The JRequest class in Joomla! has functions ready to assist with retrieving this data safely. Use JRequest::get('post') to retrieve all of the elements in the HTTP POST request as a sanitized array.

Once you have this array, you can pass it into the bind() method of JTable. Doing this will match the associated items of the array with member variables of the class. In the following example, the array is retrieved from JRequest::get('post') and immediately passed into bind().

```
if (!$row->bind( JRequest::get( 'post' ) )) {
 return JError::raiseWarning( 500, $row->getError() );
}
```

If bind() fails, you want to stop the application and explain the failure before your extension attempts to send the data. The raiseWarning() function of JError allows you to stop Joomla!, while the getError() function returns the error message stored in the JTable object.

When binding succeeds and your object is ready, call the store() function. Again, if something goes wrong, stop the application and explain why.

```
if (!$row->store()) {
 JError::raiseError(500, $row->getError() );
}
```

Note:

- If any member variables of your JTable object are null when store() is called, they are ignored by default. This allows you to update specific columns of your table, while leaving the others untouched. If you wish to override this behavior to ensure that all columns have a value, pass true into store().
- The JTable::bind() and JRequest::get() functions do not enforce data types. If you need a column to be a specific type (for instance, integer), you need to add this logic to your code before calling store().

Read

To load a specific row of the database with JTable, pass the key into the load() member function.

```
$row->load( $id );
```

This relies on the key column you specified in the second parameter of parent::__construct() when you extended JTable.

Delete

Like read(), delete() allows you to destroy a specific row in the table based on the key specified earlier.

```
$row->delete( $id );
```

If you want to delete multiple rows at once, you will need to write the query manually.