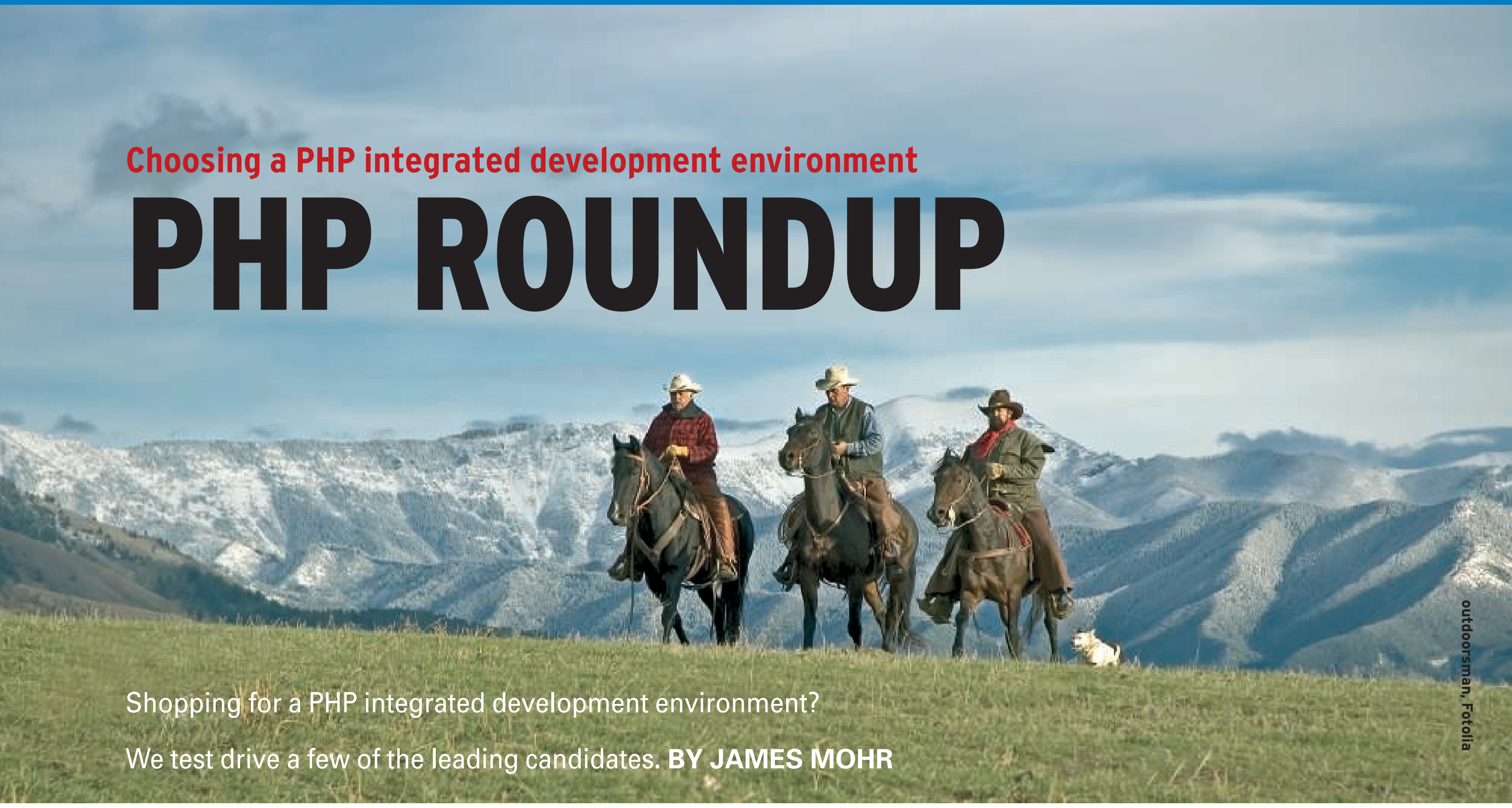


Choosing a PHP integrated development environment

PHP ROUNDUP

Shopping for a PHP integrated development environment? We test drive a few of the leading candidates. **BY JAMES MOHR**

outdoorsman, Fotolia

Long gone are the days when developers would edit source code in *vi* and run *cc* from the command line. Modern IDEs perform a wide range of functions directly from a single interface, and you never need to see a shell prompt. Several useful IDEs are available for PHP development – the right tool for you may depend on your needs and your taste for complexity. In this article, I take at closer look at some popular PHE IDEs.

I started with a small, newly created project consisting of just a handful of files. Then, I took an existing project that consisted of a PHPNuke installation, containing about 26,000 files, half of which are PHP files, but only a couple dozen of those files are changed with any frequency. Each of the products allowed me to import existing files to varying degrees. In some cases, the product simply includes *everything* in the specified directory without question. In other cases, users are prompted for the types of files to include.

Eclipse PDT

The Eclipse PHP tool PDT is released under the Eclipse Public License 1.0 [1]. Simply download the program, unpack it, and start; I downloaded version 3.3.1.1. Even though this tool is called the Eclipse PHP Development Tool, initially I didn't find any references to PHP. After checking various documents, I found that I needed to install additional packages, which was easy with the integrated software updater.

The IDE is divided into different panels. Figure 1 shows the PHP editing mode with PHP Explorer displaying the projects included in the currently active workspace. The work environment is called the workspace, and it contains one or more "perspectives." These perspectives define the layout and contents of the various windows. See the article on PHP with Eclipse elsewhere in this issue for more on working with the Eclipse user interface.

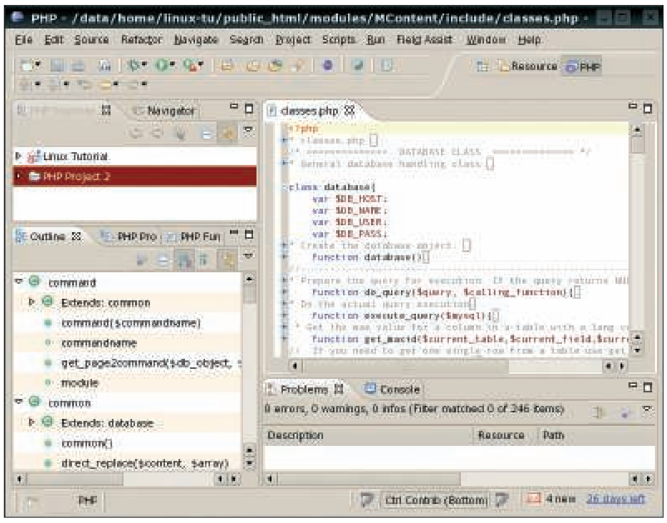
Importing my large project into Eclipse seemed to overwhelm the tool. Reaction was so slow that I thought the program had hung. Even selecting a check box sometimes took more than 10 seconds to register my selection.

One annoyance was that you cannot simply remove a file from the project. Instead, you can only delete it from the filesystem.

Built-in help searches the local help files, Google, and *eclipse.org*. Clicking the local help brought up a new copy of Firefox that connected to a local http server, which apparently starts when you launch Eclipse. Unless I dealt with 20 cookies from localhost, I couldn't get into the help.

Also, if the help page is open and you need to restart the IDE – when installing one of the many patches, for example – the existing browser window with the help pages is useless because the port is now different. Google search delivers thousands of references not related to Eclipse, and the search on the Eclipse site failed because the search criteria was not passed correctly.

Another place in which Eclipse has problems updating the GUI is in the


PHP - /data/home/linux-tu/public_html/modules/MContent/include/classes.php
File Edit Source Refactor Navigate Search Project Scripts Run Field Assist Window Help
Resource PHP
Navigator
Linux Tutorial
PHP Project 2
Outline PHP Pro PHP Fun
classes.php
Problems Console
Description Resource Path
PHP
Ctrl Contrib (Bottom)
4 new 26 days left

```
<?php
 * classes.php
/* ============== DATABASE CLASS ============== */
 * General database handling class

class database{
    var $DB_HOST;
    var $DB_NAME;
    var $DB_USER;
    var $DB_PASS;
 * Create the database object.
    function database()

 * Prepare the query for execution. If the query returns
    function do_query($query, $calling_function){
 * Do the actual query execution
    function execute_query($mysql){
 * Get the max value for a column in a table with a long v
    function get_maxid($current_table,$current_field,$curr
// If you need to get one single row from a table use get
```

```
command
  Extends: common
  command($commandname)
  commandname
  get_page2command($db_object,
  module
common
  Extends: database
  common()
  direct_replace($content, $array)
```

```
0 errors, 0 warnings, 0 infos (Filter matched 0 of 246 items)
```

Figure 1: The Eclipse PHP Development Tool.

Problems window. After correcting problems Eclipse had identified in the source code, the window did not react as expected. The *Problems* window still showed syntax errors that were no longer present, and the main IDE still marked lines with a red *X*, although they were no longer underlined in red.

One noteworthy feature is the *Task* window, where you can add a line to your code such as *// TODO: Add error handling* and then right-click the line number. Then you can add a task into the *Task* window. Later, you can double-click the entry in the task window and jump to the file and line.

One basic feature that was surprisingly missing from the default package was the ability to upload files to a remote server. However, once you figure out what term Eclipse uses for the appropriate function, you can install the *Remote System Explorer*, which allows you not only to upload the file, but "explore" the remote system. What I couldn't find was a way to select a directory but tell Eclipse to upload only the files that have changed.

The features of this product were definitely eclipsed by its user unfriendliness and repeatedly waiting for it to do what it decided to do on its own.

PDT provides many gizmos and gadgets, many of which apply primarily to hardcore programmers. If you have been using Eclipse for years to program Java or C++ applications and you are used to the product's idiosyncrasies, it is probably a useful tool.

If you are looking for an easy-to-use IDE to quickly get up and running, it is best to look elsewhere.

Zend Studio

Zend Studio [2] is an extensive and easy-to-use IDE. The ZenStudio IDE is a commercial product that sells for around US$ 254. I downloaded an evaluation copy of version 5.5.0 for this review.

Zend Studio installs from a self-extracting shell script. I was pleasantly surprised that it provides its own Java Runtime version, and the install was amazingly fast.

As you can see in Figure 2, the main interface consists of several different panels, which you can move around freely in the interface. When creating a project, you have fewer options than with other products. For example, you have a choice of directory path, but not individual files or file types. When I loaded the files, there was no problem with the number of files in my project. Everything was smooth and quick. Whether I was editing, debugging, or moving through the various trees, I noticed no performance problems.

You can configure a kind of SQL browser that can connect to multiple database and database types. Simply input the necessary connection string and it shows you all of the databases and tables to which the given user has access. The columns in the database are drag and drop, which means you can simply drop column names into your code. Right-click the table or column name, and you can show the metadata, as well as perform an arbitrary query.

When you right-click a table and select *table data*, you can see all of the data in the tables.

Zend Studio's *Inspect* window provides a display of classes, methods, and PHP functions. In the *Project* view, you see everything related to your project in all files. Despite displaying everything at once, I had no performance problems at all. The *PHP* tab lists PHP functions in commonly used groups; however, I was not able to drag the functions into the editor window as I would have expected.

Code folding works as with most applications, allowing you to fold classes, functions, and even comments; however, you cannot fold if-then blocks and similar blocks. Also, the fold location was not as visible as with other applications.

The Zend Studio code completion is perhaps the most extensive of all the products. In the editor, a small popup window appears with matching functions as you type. The list includes not only standard PHP functions, but also functions appearing other places in your project. Standard PHP functions get a fairly extensive help popup, and for your own functions, the window shows you the values expected.

Creating code snippets is straightforward, and there was a huge library of pre-existing snippets; however, I could not find any way to create my own toolbars or change existing toolbars. On the other hand, code debugging was straightforward. Unlike some other applications, Zend Studio simply worked without any additional configuration.

Zend Studio is an extensive and effective tool that is reasonably priced and worth a second look for professional use.

NuSphere PhpED

As Tom Petty put it, "The waiting is the hardest part," and this adage definitely applies to NuSphere's PhpED [3]. Even simple tasks such as saving a file were stopped in their tracks by apparent freezing of the application; I say "apparent" because running *top* was showing

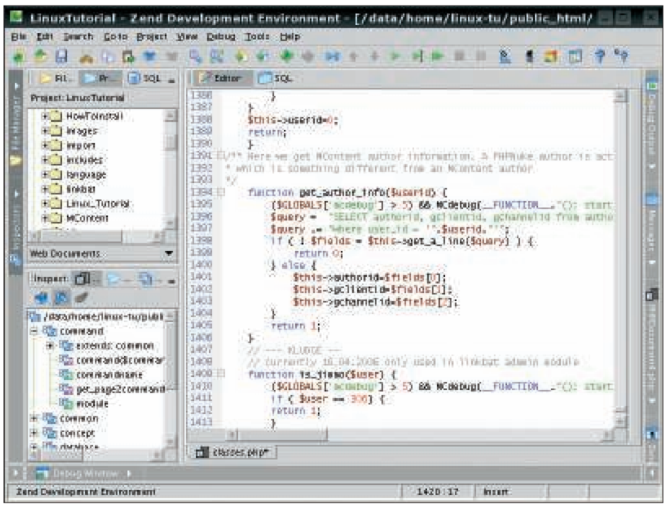
The State of the Art

Modern IDEs typically contain a common set of base features, not the least of which is an editor. In many cases, you have the choice of configuring different editors and behavior depending on the type of source code you are working with. One of the key features of IDEs is the ability to collect files into a single unit, called a *project*. You can then set various configuration options that apply each time you open any given project. Integration with a source code control tool such as Subversion or CVS is also a common feature.

A couple of useful features you typically do not find in a standard editor are code folding and syntax highlighting. Code folding allows you to compact or "fold" blocks of code to give you a better overview, particularly if you have lots of code in a single file. Typically, the first line of the block is left as is to help you identify the block, and when you need to see the entire block, you simply unfold it. With syntax highlighting, different elements of the language are displayed in different colors; for example, variables are one color, constants are another, and function names are a third.

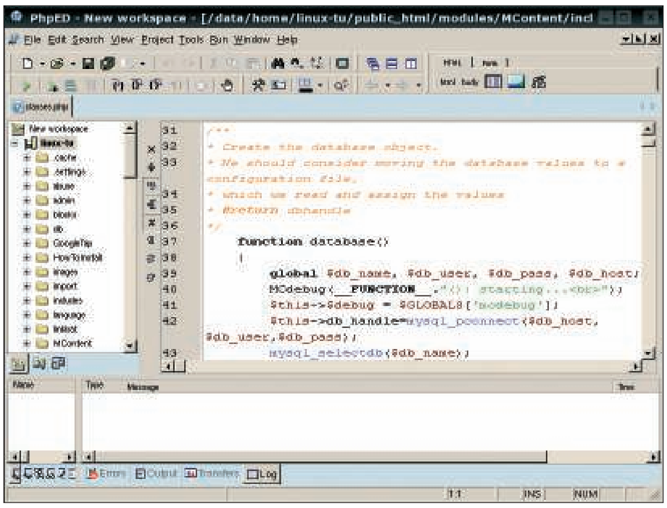
Debugging is also an important aspect of an IDE. The debugging process typically includes the ability to stop at breakpoints in your code, follow variables as their values change, and step through the program line by line. Debugging is not a feature included in all the products I tested, and in some of the products I reviewed, configuring to debug even a simple program was a project in itself.

The ability to view the structure of your documents is also a standard feature, and most IDEs provide a "class view" of files. Some views are limited to the currently active file, whereas others show additional files as well.



```
        }
    }
    $this->userid=0;
    return;
    }
/** Here we get MContent author information. A PHPNuke author is act
 * which is something different from an MContent author
 */
    function get_author_info($userid) {
        ($GLOBALS['mcdebug'] > 5) && MCdebug(__FUNCTION__."(): start
        $query =  "SELECT authorid, gclientid, gchannelid from autho
        $query .= "where user_id = '".$userid."'";
        if ( ! $fields = $this->get_a_line($query) ) {
            return 0;
        } else {
            $this->authorid=$fields[0];
            $this->gclientid=$fields[1];
            $this->gchannelid=$fields[2];
        }
        return 1;
    }
    // --- KLUDGE ---
    // currently 16.04.2006 only used in linkbot admin module
    function is_jimmo($user) {
        ($GLOBALS['mcdebug'] > 5) && MCdebug(__FUNCTION__."(): start
        if ( $user == 300) {
        return 1;
        }
```

Figure 2: The ZendStudio user interface consists of three panels.



```
/**
* Create the database object.
* We should consider moving the database values to a
configuration file,
* which we read and assign the values
* @return dbhandle
*/
    function database()
    {
        global $db_name, $db_user, $db_pass, $db_host;
        MCdebug(__FUNCTION__."(): starting...<br>");
        $this->$debug = $GLOBALS['mcdebug'];
        $this->db_handle=mysql_connect($db_host,
$db_user,$db_pass);
        mysql_selectdb($db_name);
```

Figure 3: NuSphere PhpED for Linux is missing some features.

that the program was still running, using 95+ percent of the CPU. There were no popup windows or any additional indicators of what was going on, but the program was simply not reacting.

I installed the evaluation version of PhpED version 3.3.3, which is limited to 30 days. When you download it, you can also download a patch that enables it to work with PHP 5. The current version of PhpED for Windows is PhpED 5.2, and all of the information on their site about features and functionality is for the Windows version. I couldn't find anything that described the differences between the Windows and Linux versions. Despite being two version major versions below the Windows edition, the Linux PhpED still costs US$ 299 (US$ 50 more if you want the physical media).

The product installs from a self-contained shell script. Nothing really makes PhpED stand out, and it appears to be much like other PHP IDEs, but with a different look and feel. Mostly, it felt like this was an incomplete product and the developers took an old source code tree and published the first version that compiled successfully.

I successfully imported my project by simply defining the project root as the top-level directory of the respective tree. You can define the project properties so that PhpED does *not* load specific files, but I could find no option to let me include specific files. In my case, I either ended up with a lot of extraneous files or had to add a lot of files by hand.

A lot of the windows were difficult to read. Although I have a 21" monitor and every other application works fine, PhpED does not appear to match the application font to what the system is using, and it wasn't obvious how to change this.

NuSphere's response to this problem was that they choose to compile PhpED with QT 2.3, which "is not provided by the vendor for a long time and looks like it has been discontinued," and therefore, "PhpED has no relation to this problem."

The *Code Explorer* window shows the various classes plus the methods and variables each class contains. Like most of the IDES, each of these is only displayed the first time it appears, so you need to search the code for subsequent instances.

The documentation says you can create your own menus, and I eventually found the place in the settings configuration that allowed me to do this; however, after I added an entry to simply write the current date to the error window, the application froze again for several minutes. When the application finally recovered, I had to hunt around to find where it had hidden the new menu.

When I found it and started the script, the application froze again. This time, there was a pop-up window that said "Please wait," which I did for about five minutes before the application came back to life.

In all fairness, the next time I ran the same script, the output was immediately visible; however, when I changed the script, the changes did not seem to take effect, as if the older version was being cached somewhere.

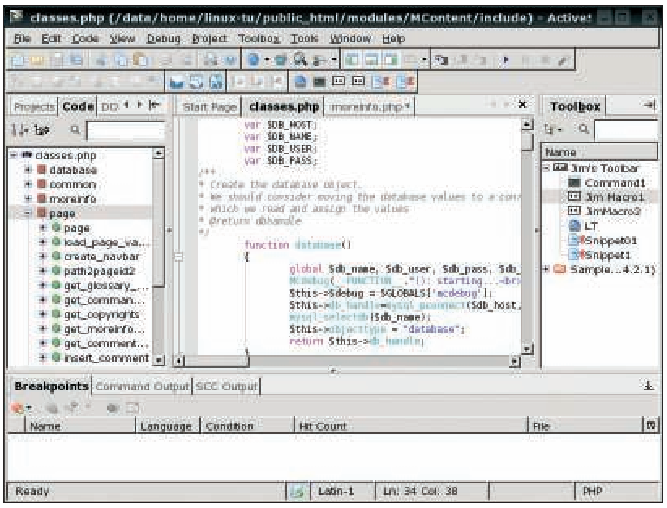
Files cannot be removed from the project but only deleted from disk. When I deleted a single file, the interface seemed to freeze. Perhaps it was updating its configuration because this one file was no longer part of the project, but it was several minutes before I could do anything else, and there were no visible indicators it was doing anything at all.

If code folding is possible, it is a well-kept secret – I found no reference to it in any menu. The NuSphere website talked about how useful this feature is, but as was so often the case, it talked about the features of the Windows version and failed to mention that the Linux version was so far behind that this particular feature was apparently missing.

PhpEd does have a couple of cool features, including an *Explorer* window, which you can use to connect to a remote machine and traverse local as well as remote filesystems (using FTP). Also, in *Code Explorer*, you can display the whole project along with all of the classes, functions, and variables in every file. Unfortunately, I found no way to search, and if you have a lot of files, it takes a while to scroll down.

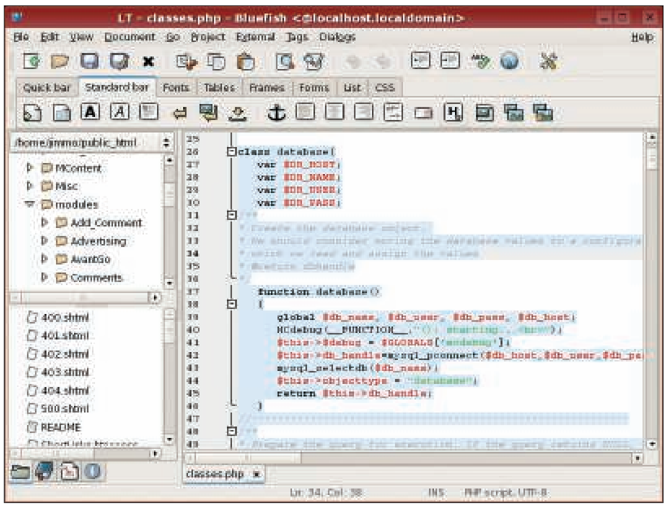
Despite some nice features, the overall appearance, behavior, and performance leave so much to be desired that it was no fun working with PhpED, even for this short test. If this were an open source project that I could download for free, I would applaud the authors and tell them to keep going. Spending hundreds of dollars for PhpED, however, is way too much.

Although my tests only covered the native Linux version, NuSphere says that the Windows version will run fine under Wine with "a few limitations." If you



```
classes.php (/data/home/linux-tu/public_html/modules/MContent/include) - Active!
        var $DB_HOST;
        var $DB_NAME;
        var $DB_USER;
        var $DB_PASS;
/**
* Create the database object.
* We should consider moving the database values to a con
* which we read and assign the values
* @return dbhandle
*/
        function database()
        {
                global $db_name, $db_user, $db_pass, $db_
                MCdebug(__FUNCTION__,"(): starting...<br
                $this->$debug = $GLOBALS['mcdebug'];
                $this->db_handle=mysql_pconnect($db_host,
                mysql_selectdb($db_name);
                $this->objecttype = "database";
                return $this->db_handle;
```

Figure 4: ActiveState's Komodo IDE in action.



```
LT - classes.php - Bluefish <@localhost.localdomain>
class database{
    var $DB_HOST;
    var $DB_NAME;
    var $DB_USER;
    var $DB_PASS;
/**
* Create the database object.
* We should consider moving the database values to a configura
* which we read and assign the values
* @return dbhandle
*/
    function database()
    {
        global $db_name, $db_user, $db_pass, $db_host;
        MCdebug(__FUNCTION__,"(): starting...<br>");
        $this->$debug = $GLOBALS['mcdebug'];
        $this->db_handle=mysql_pconnect($db_host,$db_user,$db_pa
        mysql_selectdb($db_name);
        $this->objecttype = "database";
        return $this->db_handle;
    }
//
/**
* Prepare the query for execution. If the query returns NULL
```

File Edit View Document Go Project External Tags Dialogs Help
Quick bar Standard bar Fonts Tables Frames Forms List CSS
/home/jimmo/public_html
MContent
Misc
modules
Add_Comment
Advertising
AvantGo
Comments
400.shtml
401.shtml
402.shtml
403.shtml
404.shtml
500.shtml
README
classes.php
Ln: 34, Col: 38 INS PHP script, UTF-8

Figure 5: Bluefish provides no-frills, basic funtionality.

have more applications that you need to run under Wine, this might be a viable solution, but considering all the other issues, I would not want to go through the effort of setting up Wine just to use PhpED.

ActiveState Komodo

ActiveState's Komodo [4] is a commercial product that provides everything one could want in an IDE. I have worked with ActiveState's Perl version on Windows, so I am already familiar with the quality and professionalism of ActiveState's products.

I downloaded an evaluation version of Komodo 4.0. The full version costs around US$ 295. From what I saw and read on their website, the trial version is only limited by time and not by the included features. The installation is straightforward; just download and unpack a *.tar.gz* file, then execute the included *install.sh* script.

The layout of Komodo (Figure 4) is essentially the same as other applications, with the interface broken down into multiple windows; however, I wasn't able to move the windows as I could with some applications.

Komodo comes with a number of tutorials covering a range of languages, including PHP, Ruby, Ruby on Rails, Perl, Python, and even XSLT. Simply select a tutorial based on the particular language, and Komodo will prompt you to open up a sample project with this language; I found this approach very useful.

Komodo seemed to be the fastest at loading the files from an existing directory. Unfortunately, I saw no way to select specific files to include or exclude. Furthermore, the listing in the project view mixes directories and files, as well as upper and lower case.

As with other products, the code view provides a nice overview of all of the open files in an easy-to-navigate tree. If the file contains classes, these are displayed along with their respective methods and variables.

With Komodo, the debugger "just worked," and I did not need to dig through any help files or mailing lists to figure it out. Breakpoints are simply set with a mouse click, and the interface is easy to use.

Figuring out how to upload files to a remote server was not easy. Although I eventually found a help entry labeled "Saving Files Remotely," the feature apparently does not work with entire directories.

Code snippets can be easily created through the menus, by right-clicking a location in the *Projects* view, selecting a block of text, and right-clicking it.

Double-clicking the snippet name in the *Projects* view inserts it, but if you forget where you stored it, the name will *not* appear if you search for it in the *Projects* view.

Macros function as they do in other applications; you can record keystrokes as you input them and save the sequence for later playback. Also, you can create macros from scratch if you know the correct syntax. As of this writing, macros can be written in either Python or JavaScript.

In Komodo, commands are external programs (or scripts) to which you can pass values and insert the output into the current file. As with snippets and macros, you can assign hotkeys to commands. Also, you can add commands to menus and toolbars.

You can create a toolbox that provides quick access to snippets, macros, commands, files, and even URLs. Another feature I liked was the ability to drag and drop the tabs for the open files, thereby changing their order. Other IDEs display the tabs in the order in which the files are opened so you cannot group them together in any way.

Here too, the cost might be a limiting factor if you are just a hobbyist, but professional programmers should take a closer look at Komodo.

Bluefish

Bluefish [5] is a simple text editor that implements some of the features of complex IDEs. I downloaded and installed the RPM version of Bluefish 1.0.7 from the Bluefish website and had no problems installing it.

Although it is has a wide range of features, Bluefish was cumbersome to use in comparison with other products. For example, when you create a project, something as basic as the base directory has to be input by hand. No error checking is done, so you could end up with a project path that doesn't exist.

The Bluefish GUI (Figure 5) is straightforward, but the menus offer little beyond basic functionality. Bluefish is more of an enhanced editor than a true IDE. The only information I found about debugging was how to debug the applications itself. Also, there is no integration with a version control system, nor can you upload files directly from the application.

Loading my existing files was simply a matter of defining the base directory, but I found no way to select any directories or specific kinds of files.

As I was digging through the menus for various options and features, the product crashed a few times – sometimes with a warning that it was giving up the ghost and would politely restart itself. In other cases, Bluefish died without a word.

Strangely, when I installed the “unstable” (development) version, I had no such problems.

An interesting feature is the set of pull-down menus for various languages. When you select an option, you are prompted for certain values, and then Bluefish will create the corresponding code for you based on the values you selected. Creating your own menu options is easy, but this feature is missing from the “unstable” version I later installed; in its place is a configurable snippet “tree.” Although the entries from previous versions are not included by default, a Python script exists to convert them.

The official release version I tested did not have any code folding, but code folding has been available for almost a year in the “unstable” version.

In addition to classes and functions, you can also fold comments and other blocks such as if-then.

Despite the limitations, Bluefish is probably a good tool for small development projects, no matter the language, but it still lacks many of the features one expects in a true IDE.

Quanta Plus

Quanta Plus [6] is free in terms of both beer and speech. The Quanta Plus IDE sits right in the middle in terms of features and functionality. Despite missing some of the features of other products, Quanta Plus is short, sweet, and to the point (Figure 6). Quanta Plus forms the bulk of the KDE Webdev package and is available by default any time you install KDE. As of this writing, the current version of kdewebdev is 3.5.8.

When first creating a project, you go through the *New Project* wizard and set all of the standard values like the remote server, local project root directory, and so forth. The wizard even allows you to download files from a remote server.

Also, you can pick and choose the directories and files you want to insert. Connection parameters are stored in profiles, so it is possible to upload the same project to multiple sites.

Quanta Plus lets you display the classes as well as the methods and variables, but you do not see the immediate relationship between a specific method or variable and the class where it is used. Instead, the functions and variables are listed separately from the classes.

On the other hand, one nice touch is that all of the variable instances – not just the first one – are listed, making it easy to find every place where a particular variable is used.

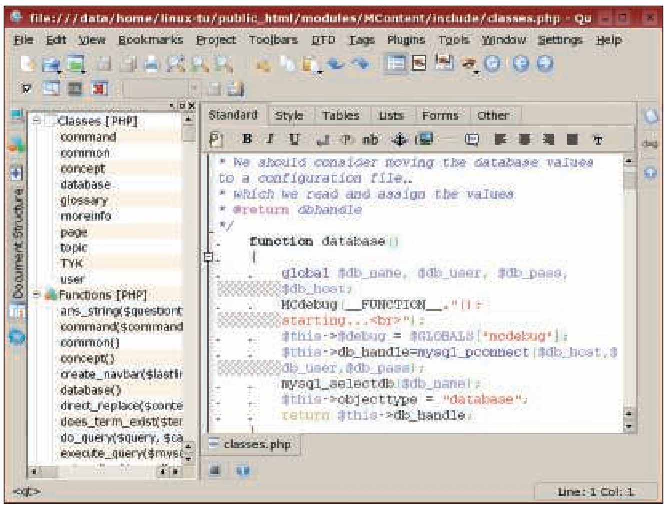
As with other applications, pressing Ctrl + Shift brings up a list of functions; however, the “hints” provided for the function seem to be lacking in a few cases.

As for code folding, Quanta Plus seems to be at the head of the class. Every element, right down to if-then statements and comments, can be folded. The folding also seemed to be more intelligent than in some of the other products. In my code, I have cases where *if* statements are written on just one or two lines, in which case Quanta Plus leaves these statements as is.

Another aspect I really like about Quanta Plus is the configurability. Creating your own commands, hotkeys, snippets, and so forth is pretty standard, and I found Quanta Plus to be the simplest and easiest to use in this regard. Toolbars can be quickly added. You can add buttons for all of the various functions and snippets to the toolbar and then pack it up and send it to other developers so they are using the same set of tools.

One reason I like Quanta Plus the best is my experience with the developers of the product. When I reported a bug in the user interface, the primary developer confirmed the problem and uploaded a patch in less than an hour. Although patches typically take longer to get online, the speed at which some patches do get online is amazing.

Debugging is an area in which I think Quanta Plus falls short. Definitely it was not as straightforward as in Komodo. Debugging must be activated explicitly for the given project. If you don’t want to spend the money, and you need to jump in quickly to get your project moving, Quanta Plus is my recommendation.



```
* We should consider moving the database values
to a configuration file..
* which we read and assign the values
* @return dbhandle
*/
    function database()
    {
        global $db_name, $db_user, $db_pass,
        $db_host;
        MCdebug(__FUNCTION__."():
        starting...<br>");
        $this->$debug = $GLOBALS['nodebug'];
        $this->db_handle=mysql_pconnect($db_host,$
        db_user,$db_pass);
        mysql_selectdb($db_name);
        $this->objecttype = "database";
        return $this->db_handle;
```

Figure 6: Quanta Plus offers advanced features with no-cost open source licensing.

Conclusions

Despite my preference, QuantaPlus may not satisfy everyone’s needs – it lacks some of the features that other products have, and the look and feel is different.

While at the high end in terms of features, Eclipse had performance problems that made it difficult to use; however, I have hopes this will change. Zend Studio and Komodo are commercial products that are still worth considering if you develop professionally. ■

INFO

[1] Eclipse PHP Development Tool (PDT): *http://www.eclipse.org/pdt*

[2] ZendStudio: *http://www.zend.com/products/studio*

[3] NuSphere PhpED: *http://www.nusphere.com/products/phped.htm*

[4] ActiveState Komodo: *http://www.activestate.com/Products/komodo_ide*

[5] Bluefish: *http://bluefish.openoffice.nl*

[6] Quanta Plus: *http://kdewebdev.org*