Benefits of using namespace in zend 2

What are namespaces?

From the PHP [manual](http://php.net/manual/en/language.namespaces.rationale.php):

*namespaces are a way of encapsulating items*

Hardly the most useful of definitions, but it's a starting point! A namespace is a way of grouping code that exists across multiple files without having a naming collision. That is, you can have the same named class in two different places if they are encapsulated within namespaces.

Of course the way that ZF1 solves this problem is with Extra\_Long\_Classnames which are obviously unique. Namespaces allow us to reference the classname by the important bit (the last section and not have to carry the meta data (where it lives) in every use of the class.

That is, namespaces allow us to:

* combine libraries with the same classnames
* avoid very long classnames
* organise our code easily

Note that namespaces do not just affect classes. They also affect functions and constants.

Defining a namespace

There is a new keyword called namespace which is used to declare a namespace for a file. This is file-wide:

namespace MyDbStatement;

class Sqlsrv extends AbstractStatement

{

}

Note that we can have multiple sub-namespaces, separated by the backslash. When we extend the Sqlsrv class the extended class is within the namespace too, but not in the same file.

We can then use the class like this:

$stmt = new MyDbStatementSqlsrv();

You can have multiple namespaces in a file, but the first namespace must be the first line of code in the file (except comments and declare statements).

Working within the same namespace

When you are working within the same namespace, then any unqualified functions and classes will be resolved to the current namespace. i.e:

namespace MyDbStatement;

function testSqlsrv()

{

$stmt = new Sqlsrv();

}

In this case, the SqlSrv class is in the namespace MyDbStatement.

Namespace importing: the use keyword

We can *import* a namespace into a different file using the use keyword.

namespace MyApplication;

use MyDbStatement;

$stmt = new StatementSqlsrv();

Note that we don't use a leading in the new statement as we are using a qualified namespace, not a fully-qualified one.

or you can import a specific class from a namespace;

namespace MyApplication;

use MyDbStatementSqlsrv;

$stmt = new Sqlsrv();

It follows you can use multiple use statements:

use MyDbStatement;

use MyDbAdapter;

$stmt = new StatementSqlsrv();

$adapter = new AdapterSqlsrv();

You cannot do this though:

use MyDbStatementSqlsrv;

use MyDbAdapterSqlsrv;

$stmt = new Sqlsrv();

$adapter = new Sqlsrv();

as clearly PHP cannot resolve which Sqlsrv class to instantiate.

You can also *alias* namespaces. This allows us to reference a long namespace with a shorter name or to import two namespaces having the same name and give them different names.

use MyDbStatementSqlsrv as DbStatement;

use MyDbAdapterSqlsrv as DbAdapter;

$stmt = new DbStatement();

$adapter = new DbAdapter();

This also allows you to write code the focusses on the functionality of the class rather than the specific type. For example, we could start using the Mysqli versions of the statement and adapter by just changing the usestatements.

The \_\_NAMESPACE\_\_ constant

The constant \_\_NAMESPACE\_\_ provides the current namespace name. In the global space it will be an empty string.

Namespace resolution

This bit is really important!

An *unqualified class name* is resolved in this order:

1. If there is an import statement that aliases another name to this class name, then the alias is applied.
2. Otherwise the current namespace is applied.

An *unqualified function name* has different rules:

1. The current namespace is prepended to the function name.
2. If the function names doesn't exists in the current namespace, then a global function name is used if it exists.

This means that within a namespace'd file, you can do:

$date = date('Y-m-d');

but not:

$datetime = new DateTime();

Instead, you have to use:

$datetime = new \DateTime();

Conclusion

That's really all you need to know to use namespaces in a PHP application. They aren't so hard really.