Simple logging of ZF2 exceptions

I recently had a problem with a ZF2 based website where users were reporting seeing the error page displayed, but I couldn't reproduce in testing. To find this problem I decided to log every exception to a file so I could then go back and work out what was happening. In a standard ZF2 application, the easiest way to do this is to add a listener to the 'dispatch.error' event and log using `ZendLog`.

To do this, I started with the Application's `Module` class and added an event listener:

```
public function onBootstrap($e)
{
    $eventManager = $e->getApplication()->getEventManager();
    $eventManager->attach('dispatch.error', function($event){
        $exception = $event->getResult()->exception;
        if ($exception) {
            $sm = $event->getApplication()->getServiceManager();
            $service = $sm->get('ApplicationServiceErrorHandling');
            $service->logException($exception);
        }
    });
}
```

This code attaches an anonymous function to the 'dispatch.error' event which retrieves the exception from the event's result and passes it to the `logException()` method in an `ErrorHandling` class. We retrieve ErrorHandling from the service manager which allows us to inject an instance of `ZendLog` into it:

```
public function getServiceConfig()
{
    return array(
        'factories' => array(
            'ApplicationServiceErrorHandling' =>  function($sm) {
                $logger = $sm->get('ZendLog');
                $service = new ErrorHandlingService($logger);
                return $service;
            },
            'ZendLog' => function ($sm) {
                $filename = 'log_' . date('F') . '.txt';
                $log = new Logger();
                $writer = new LogWriterStream('./data/logs/' .
$filename);
```

```
                    $log->addWriter($writer);

                    return $log;
                },
            ),
        );
    }
```

There's obviously a few use statements at the top of the file for this to work:

```
use ApplicationServiceErrorHandling as ErrorHandlingService;
use ZendLogLogger;
use ZendLogWriterStream as LogWriterStream;
```

The logging itself is done within the `ErrorHandling` class:

```
namespace ApplicationService;

class ErrorHandling
{
    protected $logger;

    function __construct($logger)
    {
        $this->logger = $logger;
    }

    function logException(Exception $e)
    {
        $trace = $e->getTraceAsString();
        $i = 1;
        do {
            $messages[] = $i++ . ": " . $e->getMessage();
        } while ($e = $e->getPrevious());

        $log = "Exception:n" . implode("n", $messages);
        $log .= "nTrace:n" . $trace;

        $this->logger->err($log);
    }
}
```

The `logException` method simply creates a string containing the exception's message along with any previous exception messages and the trace. We then call the `Log`'s `err` method to store the log and can peruse at our leisure.

Update: I have updated the `logException` method to use a do..while() loop as it's neater and doesn't cause a an out-of-memory error that the previous code did. That is, it's a good idea to reuse the same variable when calling `getPrevious()`!