TYPO3 Core - Feature #91222

Make dependency injection possible for errorPhpClassFQCN

2020-04-28 14:41 - Christian Futterlieb

Status:	New	**Start date:**	2020-04-28
Priority:	Should have	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Link Handling, Site Handling & Routing	**Estimated time:**	0.00 hour
Target version:			
PHP Version:		**Complexity:**	
Tags:		**Sprint Focus:**	

Description

Because TYPO3\CMS\Core\Site\Entity\Site::getErrorHandler() passes two arguments to the constructor of classes declared as errorPhpClassFQCN, dependency injection is not possible.

I guess, the solution should be to change the method PageErrorHandlerInterface::handlePageError() to accept the $statusCode and $errorHandlerConfiguration. Or add a new interface method for compatibility/deprecation reasons.

Additionally, a new method Site::getErrorHandlerConfiguration(int $status) could remove the binding of creating and configuring an errorPhpClassFQCN.

What are your thoughts? I'd also be happy to provide some code for this.

History

#1 - 2020-04-29 10:07 - Benjamin Franzke

We can not change the method as PageErrorHandlerInterface is an interface and we can't add method arguments – not even optional/nullable ones. We'd need to add a new StatusAndCongfigurationAwarePageErrorHandlerInterface with a new method.

I think it's not that nice for the core code to have to handle mutliple error handler interface types…

Still it is a good Idea to support dependency injection!

I think I'd rather prefer a new handler type "Factory" (additonal to "PHP" which wouldn't be removed).
The factory would be a container service that implements `PageErrorHandlerFactoryInterface`.

The interface for such factories would look something like this:

```
interface PageErrorHandlerFactoryInterface {
    public function createErrorHandler(
        int $statusCode,
        ?array $errorHandlerConfiguration = null
    ): PageErrorHandlerInterface;
}
```

This factory could then instantiate an own implementation of an error handler: That could either be a DI prototype class (shared: false) that contains a setter for $statusCode + $configuration (whatever you actually need) or simply an anonymous class…

Here is an example for an anonymous class:

```
class MyPageErrorHandlerFactory implements PageErrorHandlerFactoryInterface {

    public function __construct(MyDependency $myDependency) {
        $this->myDependency = $myDependency;
    }

    public function createErrorHandler(
        int $statusCode,
        ?array $errorHandlerConfiguration = null
    ): PageErrorHandlerInterface {
        return new class($myDependency, $statusCode, $errorHandlerConfiguration) implements
PageErrorHandlerInterface {
            public function __construct(MyDependency $myDependency, int $statusCode, ?array
$errorHandlerConfiguration = null) {
                $this->myDependency = $myDependency;
```

```
                $this->status = $statusCode;
            }
            public function handlePageError(ServerRequestInterface $request, string $message, array $reasons =
 []): ResponseInterface {
                // Do something using $this->myDependency here
            }
        }
    }
}
```

The page configuration would looks something like:

```
errorHandling:
  -
    errorCode: '404'
    errorHandler: Factory
    errorFactoryService: Vendor\ExtensionName\ErrorHandlers\MyPageErrorHandlerFactory
    # or maybe rather
    errorHandlerFactoryService: Vendor\ExtensionName\ErrorHandlers\MyPageErrorHandlerFactory
    # or just (to be discussed)
    errorHandlerFactory: Vendor\ExtensionName\ErrorHandlers\MyPageErrorHandlerFactory
```

Of course, this is just my taste, I'll ask other core devs whether they would be fine with such interface.

Do you want to try to start coding on that, or should I do?

#2 - 2020-04-29 11:24 - Christian Futterlieb

I like your idea, it seems to be a very elegant solution. Imo, it could even deprecate/supersede the current 'PHP' errorHandler, because having two methods of 'add-your-own-code' might be a bit too much.

> Do you want to try to start coding on that, or should I do?

When your colleagues are fine with it, i'll go when you say so ;-) That work would be considered a new feature and go to master only, right?