Welcome

Extension and Modularization of VuFind: Ideas – and Questions

Hajo Seng, SUB Hamburg
Hamburg, 1.10.2020
Why extending VuFind? Isn’t it enough?

specific usecases

e.g. a „regions“ catalogue
Why extending VuFind? Isn‘t it enough?

**specific usecases**
- e.g. a „regions“ catalogue

**real extensions of functionality**
- e.g. connected delivery services
Why extending VuFind? Isn‘t it enough?

specific usecases
  e.g. a „regions“ catalogue

real extensions of functionality
  e.g. connected delivery services

particular requirements
  e.g. a special year facet
Why using a modular design?

E.g. beluga-core consortium:

- consortium of libraries with and without programming powers
  - yet: 7 libraries in northern Germany
- aim: adjust catalogue frontend to specific needs
- shared development of VuFind-extensions
- sharing features with other libraries

AIM:
- local functionality: creating a module
- global functionality: preparing a pullrequest at VuFind
  (and creating a temporary module)
Extending modules: an approach

modules: extending namespaces

themes: using mixins
naming: <module>-<template>.phtml
basic mixin with empty templates

languages, configs: using parents

others: database, composer, ...
# Interfaces for module extensions

<table>
<thead>
<tr>
<th>Interface</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td>convention: themes &amp; modules have the same name</td>
</tr>
<tr>
<td>module.config.php</td>
<td>native inheritance</td>
</tr>
<tr>
<td>ViewHelper</td>
<td>naming: <code>&lt;module&gt;\View\Helper\&lt;theme&gt;</code></td>
</tr>
<tr>
<td>AjaxHandler</td>
<td>inheritance possible, often not necessary</td>
</tr>
<tr>
<td>Controller</td>
<td>inheritance possible, often not necessary</td>
</tr>
<tr>
<td>classes / objects</td>
<td>discussion</td>
</tr>
<tr>
<td>PluginManagers</td>
<td>discussion</td>
</tr>
</tbody>
</table>
Stacking modules

There are some popular objects which are overridden by several modules. The order of modules is important. Use-declaration has to be changed if modules are plugged in or out. Extension-chains can be avoided in AjaxHandlers, Controllers, ViewHelpers. But not everywhere.
Stacking modules: PluginManagers

PluginManagers can’t be overridden
doubled aliases & factories
Stacking modules: PluginManagers

PluginManagers can’t be overridden

Is there a better solution with VuFind 7?
Solutions and questions

themes
mixins, naming-conventions

language- & configurationfiles
inclusion of parents

modules
native inheritance

module.config.php
naming-conventions

ViewHelpers
avoid inheritance

AjaxHandlers, Controllers
manipulate use-declarations

classes / objects
override (and double) properties

PluginManagers

others (database, composer)
additional configuration snippets
Solutions and questions

AIM: pluggable extensions

modulestack: dependencies of stacked objects:
look after the order of modules
change use-declarations
not pluggable (but almost)

translation- and configfiles: change parent-declarations
not pluggable (no source code affected)

themes: completely pluggable

others: another topic ... (particularly composer-extensions)
Thank you very much

Hajo Seng
Von-Melle-Park 3
20146 Hamburg

040 / 42838-8336
hajo.seng@sub.uni-hamburg.de

www.sub.uni-hamburg.de
facebook.com/stabihh
twitter.com/stabihh

www.beluga-core.de
github.com/beluga-core/core