TYPO3 Core - Story #64274

Add new Plugin registration

2015-01-14 12:56 - Mathias Schreiber

Status:	Accepted	**Start date:**	2015-01-14
Priority:	Should have	**Due date:**	
Assignee:	Mathias Schreiber	**% Done:**	0%
Category:	System/Bootstrap/Configuration	**Estimated time:**	0.00 hour
Target version:	Candidate for Major Version		
TYPO3 Version:	7	**Tags:**	
PHP Version:	5.5	**Sprint Focus:**	

Description

Registering a non-extbase plugin currently needs two things:

- Add the TCA necessary by calling

  `\TYPO3\CMS\Core\Utility\ExtensionManagementUtility::addPlugin`

- Add the typoscript by calling

  `\TYPO3\CMS\Core\Utility\ExtensionManagementUtility::addPItoST43`

addPItoST43 adds typoscript in this form:

```
plugin.tx_extensionkey_suffix = USER
plugin.tx_extensionkey_suffix.userFunc = tx_extensionkey_suffix->main
```

When you want to use namespaced code (and I say that's what we all want) you need to manually supply a classmap in this form:

```
tx_extensionkey_suffix => '\Vendorname\Extensionname\Controller\Class'
```

I propose a new method to register non-extbase plugins that enforces stronger defaults and reduces the clutter in the Typoscript Obejct Browser.

New method:

```
static public function addPluginTyposcript($FQCN, $methodName = 'render')
```

See the example:

```
static public function addPluginTyposcript(\MyAgency\MyExtension\Cached\Plugin\MyPlugin::class, 'd
ispatch')
static public function addPluginTyposcript(\MyAgency\MyExtension\UnCached\Plugin\MyPlugin::class,
'dispatch')
static public function addPluginTyposcript(\MyAgency\MyExtension\Cached\Menu\MyMenu::class, 'rende
r')
static public function addPluginTyposcript(\MyAgency\MyExtension\Cached\CType\MyCType::class, 'ren
der')
static public function addPluginTyposcript(\MyAgency\MyExtension\Cached\Header\MyHeader::class, 'r
ender')
```

As you can see the FQCN (Fully qualified class name) holds a strong default for registering a plugin in the frontend.
Let's take the example above apart:

- \MyAgency
  - the Vendor name
- \MyExtension
  - Your extension key

- \Cached
    - defines whether a plugin runs cached or uncached (just for TS registering, you can re-define it via TS, possible values are:
        - Cached
        - Uncached
- \Plugin
    - Determines the type, possible values are:
        - ~~Plugin~~
        - Menu
        - CType
        - Header
- \MyPlugin
    - Your class name

We will drop support for "just include library", because the design is just broken.
All TS get's added to page.1000, which is problematic if

- you have multiple of these plugins installed
- your root object simply has another name than page

Typoscript result

I propose to add more "dots" to the TS.
Possible result:

```
plugin.MyAgency.MyExtension.Plugin.MyPlugin = USER
plugin.MyAgency.MyExtension.Plugin.MyPlugin.userFunc = \MyAgency\MyExtension\Cached\Plugin\MyPlugi
n->dispatch
plugin.MyAgency.MyExtension.Menu.MyMenu = USER
plugin.MyAgency.MyExtension.Menu.MyMenu.userFunc = \MyAgency\MyExtension\Cached\Menu\MyMenu->rende
r
```

This will clean up the Typoscript Object browser.

tt_content.CType result

We could also derive the TCA necessary from this classes.
The following fields in tt_content are affected:

- CType
- ~~list_type~~
- menu_type
- header_layout

Storing values in these fields needs to comply to the TS being generated.
So the resulting value from the example would be:

```
MyAgency.MyExtension.Plugin.MyPlugin
MyAgency.MyExtension.Menu.MyMenu
```

When selecting values in the database this comes in handy because you can now use proper wildcards to isolate specific extensions.

In order to generate the label for the field in the BE we would use a locallang.xlf hit for the same identifier.

Registering multiple Plugins/CTypes

Currently TYPO3 loops through all registered plugins and adds the TS on demand (in every hit).
Additionally, it's painful if you have an extension that registers, say, 50 CTypes.
The idea would be to look up all CTypes automatically and cache this information.
This way an extension would just need to "configure" something like "I do supply UserFunctions".
If there are cached registrations, use those.
If no cached registrations are in place, try to look them up via the filesystem (using the logic described above) and build up the caches.

History

#1 - 2015-01-14 12:56 - Mathias Schreiber

- PHP Version set to 5.5

#2 - 2015-01-14 12:59 - Andreas Wolf

Looks good :-)

The only thing that bugs me after reading it once again is the cached/uncached stuff. I’m not sure if the class namespace is really the right place to have such information. Having two methods registerCachedPlugin/registerUncachedPlugin or a separate caching parameter would be better IMHO.

#3 - 2015-01-14 13:02 - Benni Mack

What I would like to see is

1) The plugin registration should be unified, also working for extbase extensions
2) I think a strong default should be not using "plugin" (CType list, subtype list_type) anymore but CTypes directly for the future.
3) I would love to have an auto-registration (similar to EXT:autoloader) instead of adding this to ext_localconf.

I can write down a blueprint for that

#4 - 2015-01-14 13:23 - Mathias Schreiber

- Description updated

#5 - 2015-01-14 13:37 - Carsten Bleicker

I need 2 Method calls?
\TYPO3\CMS\Core\Utility\ExtensionManagementUtility::addPlugin
\TYPO3\CMS\Core\Utility\ExtensionManagementUtility::addPluginTyposcript

if so, why not just one?

#6 - 2015-01-14 13:48 - Mathias Schreiber

As written above:
One thing is TS, the other thing is TCA related.

It could be unified, that's the plan.

#7 - 2015-01-14 13:53 - Mathias Schreiber

- Description updated

#8 - 2015-01-14 14:00 - Mathias Schreiber

- Description updated

#9 - 2015-01-14 14:10 - Markus Klein

Extbase only supports plugins and and ctype, but not headers and menu.

I'm not sure if the change of the TS syntax is really possible at the current stage. This would maybe work out for third party extensions, but it will break IndexedSearch which would be the first candidate needing this API.

#10 - 2015-01-14 14:12 - Mathias Schreiber

I gave indexed_seach some thought and I think it would be easier to supply a classmap for IS only.

#11 - 2015-01-15 16:14 - Mathias Schreiber

- Description updated

#12 - 2015-01-16 08:42 - Daniel Siepmann

Thanks for the nice topic.

I would prefer to follow Benjamin Mack:

> 3) I would love to have an auto-registration (similar to EXT:autoloader) instead of adding this to ext_localconf.

I don't want to write code at all to do this obvious things. I just want to place my code, following the strong defaults and conventions, and it's there.

I think that's pretty easy for non-extbase plugins. As you don't need any further configuration like Extbase does.

I don't think you can handle both ways the same. Extbase just needs, at the moment, further configuration which, in my opinion, can't be fetched from existing sources.
And that's mostly fine. Of course it would be better to have the same principle and way for both. But Extbase is not the same. It itself can follow the above way. But it's not TYPO3, it's Extbase, a PHP Framework inside of TYPO3 CMS.
If you stick to it, you have to follow it.

#13 - 2015-01-16 13:02 - Markus Klein

Well we can make this auto-registration based, but that would require:

- Having the information whether a plugin is cached or uncached encoded into the namespace
- The method to call has to be fixed to some value.

#14 - 2015-01-16 13:06 - Mathias Schreiber

Markus Klein wrote:

> Well we can make this auto-registration based, but that would require:
>
> - Having the information whether a plugin is cached or uncached encoded into the namespace

Or my having something like static public runScriptUncached = true;
But that would involve calling the class itself while building up the caches.

> - The method to call has to be fixed to some value.

I discussed this with CK yesterday.
He's in favor of having an explicit registration.
Maybe do a hangout with people interested.

#15 - 2015-01-16 13:21 - Markus Klein

Mathias Schreiber wrote:

> Markus Klein wrote:
>
> > Well we can make this auto-registration based, but that would require:
> >
> > - Having the information whether a plugin is cached or uncached encoded into the namespace
>
> Or my having something like static public runScriptUncached = true;
> But that would involve calling the class itself while building up the caches.

A clear no-go as this would open up things to break again during bootstrap. Imagine a constructor which does some nasty stuff. While bootstrap loads all the ext_localconf files the class would be instantiated and would therefore crash the whole instance.

> > - The method to call has to be fixed to some value.
>
> I discussed this with CK yesterday.
> He's in favor of having an explicit registration.
> Maybe do a hangout with people interested.

I guess I prefer explicit over implicit too in this case, although it is hard to draw the line. Implicit is better for avoiding configuration hassle, but explicit is better for emphasizing relationships/dependencies. Implicit as also more error-prone and harder to debug. Consider a typo in the folder name for instance. Your plugins will never show up and you've no clue why and even worse, the system can't help you out, as it simply doesn't know you were intending to add a plugin. If you have that explicitly, the system can shout: Hey I can't find that plugin you want to add. You get a clue where to start searching for the issue.

#16 - 2015-01-16 15:23 - Mathias Schreiber

Markus Klein wrote:

> A clear no-go as this would open up things to break again during bootstrap.

Agreed.

> I guess I prefer explicit over implicit too in this case, although it is hard to draw the line.

Actually no.
Explicit is easier to use AND to implement, less error prone and simply faster.

So the challenge is to make the process of registration as simple as possible and short as possible.
This includes a short, easy to read, expressive syntax.

#17 - 2015-01-16 17:25 - Daniel Siepmann

Just to bring in some ideas from other projects for the syntax. That's what I imagine could be the DSL for a Ruby Project:

```
TYPO3.extension.config do
    plugin :news
    plugin :admin
end
```

That can be the same as for Extbase. The above is without.

```
TYPO3.extension.config do
    plugin :news, cached: [ :list, :detail ], non_cached: [ :search, :result ]
    plugin :admin, non_cached: [ :edit, :new, :update ]
end
```

#18 - 2015-01-23 11:46 - Andreas Wolf

Daniel Siepmann wrote:

> That can be the same as for Extbase. The above is without.
> [...]

Nice idea, however I think this form suits a bit better:

```
TYPO3.extension.config do
    plugin :news { cached: [ list, detail ], uncached: [ search, :result ] }
    plugin :admin { uncached: [ :edit, :new, :update ] }
end
```

So this is a more JSON-style syntax, to make it more clear that the cached/uncached lists are in fact properties of the plugin; that's a point I missed in Daniel's suggestion. Apart from that, nice thing :)

#19 - 2015-02-28 17:34 - Benni Mack

- Target version changed from 7.1 (Cleanup) to 7.2 (Frontend)

#20 - 2015-06-08 14:21 - Markus Klein

- Status changed from Needs Feedback to Accepted

- Target version changed from 7.2 (Frontend) to 7.3 (Packages)

#21 - 2015-06-08 15:15 - Markus Klein

After discussing this extensively with Mathias Schreiber again, I summarize:

- we want autoregistration (real life usecase: saving lots of typing when adding 55 CEs)
- Location in extension is Classes/<Plugin|Ctype|Menu|Header>/<Cached|Uncached>/<ClassName>.php
- Method called in class is "render()"

The TypoScript and TCA registration will be autogenerated, no need to call addPlugin and addPItoST43 anymore.

The TypoScript changes proposed above will be omitted for the time being.

#22 - 2015-06-08 16:02 - Thomas Maroschik

I have several issues with the suggested solution:

Implementing this through autoregistration introduces a lot of complexity and overhead into requests: like inflecting the class name from a filename is cumbersome and error prone. further the "buildup" time of CMS increases, when loading fails it doesn't fail with a error message but just doesn't work, etc. further class loading is more and more a duty of composer and the CMS shouldn't know about the implementation details of composer, so it should just load classes transparently from what location ever. caching the information introduces it's own issues, like when to invalidate, when to rescan. for the system it's hard to detect changes automatically. debugging why a certain plugin doesn't load can sometimes take hours where the time debugging exceeds the time saved typing an explicit registration by far. I experienced this myself in other former TYPO3 products as biggest productivity killer.

So I think a better low tech solution would be to simply register this plugins in a file (maybe like the new TCA config files) where the plugin name is the key and the value is an array consisting of the full classname and the method to call. TS and tt_content TCA could still be fed from this. and the large array simply be cached in configuration cache.

We could provide a cli command that scans your extension for those classes and registers them then in a configuration file to make the life of a developer a bit easier. Further the developer can then go on and check the result of the operation. This command could also do all kinds of configuration checks and provide valuable information to the developer. This would be efficient, because the plugin information only changes on "commit time" and not during "runtime".

#23 - 2015-06-08 16:26 - Mathias Brodala

Just my two cents: Tim Lochmüller has implemented something like this with his autoloader extension.

#24 - 2015-06-08 16:42 - Patrick Broens

IMHO the whole description of this issue is wrong. You do not need addPItoST43 to register a userfunc. You can add the TypoScript yourself if you have added a CType using addPlugin. I don't even want to use addPItoST43, because it is an old magical thingy nobody understands. The proposed replacement of css_styled_content is not using this, but is using data processors from the content object FLUIDTEMPLATE, which makes this registration of classes redundant.

That does not mean we need a replacement for addPItoST43, but keep in mind there are other ways to handle data as well, as the CSC replacement is doing with data processors. They need to be assigned anyway, without magic.

#25 - 2015-06-08 16:54 - Markus Klein

Well, of course for Ctypes this "new" API will not be relevant anymore, due to new "CSC". But we still need a way to register namespaced, pibased plugins. Nothing more or less.
The discussion about the future concept solely grew out of that need.

#26 - 2015-06-08 22:06 - Christian Kuhn

I read through all comments and some of the involved code, but still need to think about that a bit longer.

At the moment, I think that an "autoregister" by putting some classes at magic places is not my favorite. I've tried that with the new toolbar registration and threw away the implementation after realizing all the magic, assumptions and ugly code the system has to go through and the hell dev's tap in during debugging if it does not work - and it never works at first try. So, I'm fully on Thomas side at this point.

Looking specifically at addPItoST43(), I'm mostly thinking about just deprecating it without substitution: It doesn't provide much value beside it puts stuff into a convention namespace - but it extracts this magically from the input parameters, and that is exactly what is hitting us currently. For indexed_search, we could just add the TS on our own and not use the method anymore. Maybe that is the most straight and simple solution ahead - at least the ugly alias could be kicked this way.

#27 - 2015-06-08 23:17 - Markus Klein

After all that fruitful discussion, I started digging a lot deeper and would suggest this solution:

Each extension can define the following files in the Configuration/ folder:

- Plugins.php
- ContentElements.php
- HeaderLayouts.php
- Menus.php

Each of them is supposed to return an array, which contains all Plugins/Menus/... including configuration.
Example

```
<?php
return [
    1433786303 => [
        'handler' => \TYPO3\CMS\IndexedSearch\Controller\SearchFormController::class,
        'cached' => FALSE,
        'icon' => '',
        'label' => ''
    ]
];
```

Where 'icon' and 'label' are optional and will be loaded from default locations, if not specified.

I inspected all places which depend on the methods addPlugin and addPItoST43 being called, which are luckily only three.

Those configuration files are loaded at different places then:

1.) \TYPO3\CMS\Core\Utility\ExtensionManagementUtility::buildBaseTcaFromSingleFiles
The TCA (currently done by addPlugin) is added here just before the TCA/Overrides are executed
=> The TCA changes are cached

2.) \TYPO3\CMS\Core\TypoScript\TemplateService::addDefaultTypoScript
The TypoScript is added and will be cached as well.
(former addPItoST43)

3.) \TYPO3\CMS\Core\TypoScript\Parser\TypoScriptParser::checkIncludeLines:858
Also a place which is influenced by plugin TS.
Here the TS needs to be added as well.
(former addPItoST43)

The benefit is a dedicated, explicit configuration, which is only evaluated when needed and is fully cached.
(Additionally, if we embrace a standard for PHP class placement, a dedicated tool coult generate these config files automatically.)

#28 - 2015-06-08 23:23 - Christian Kuhn

Markus, this is going into a sane direction, but it also opens a relatively huge topic: "Finally get configuration handling done". Let us give some time here and evolve that. Do we have enough person power to tackle this thing in 7?

If we open this pit, we need to have a look at all main registrations of std api's in the core ... this is a rather huge part.

#29 - 2015-06-08 23:44 - Markus Klein

Well, I guess it is the right direction, but maybe too early. ;-)

#30 - 2015-06-15 17:08 - Benni Mack

- Target version changed from 7.3 (Packages) to 7.4 (Backend)

#31 - 2015-07-23 15:51 - Benni Mack

- Category changed from Backend API to 1595

- Sprint Focus set to Stabilization Sprint

#32 - 2015-07-28 17:39 - Benni Mack

- Target version changed from 7.4 (Backend) to 7.5

- Sprint Focus deleted (Stabilization Sprint)

#33 - 2015-09-24 07:50 - Benni Mack

- Target version changed from 7.5 to 8 LTS

#34 - 2017-03-28 22:54 - Benni Mack

- Target version changed from 8 LTS to Candidate for Major Version