

Services are very badly documented throughout the TYPO3 documentation, hence the idea of making a presentation about them and improve the situation.

## Services in TYPO3?

- introduced in 3.6
- used for authentication (FE, BE)
- used in the DAM
- part of the core
- a type of extension

cobweb

### Usefulness of services

- a generic way of implementing features that may have many variants
- allows differentiation for special cases
- available both FE and BE

In many ways services try to provide solutions for OO features that were missing at the time in PHP: factory pattern, singleton pattern, polymorphism, exception handling. This means a lot of code may seem useless or obsolete these days, but the whole concept of services is still useful because it went beyond that and added nice extra features.

Example:

-Authentication services:

-available for FE and BE

-Many variants exists for different authentication schemes: LDAP, etc.

-File meta data extraction services:

-Used by the DAM

-Variants for each type of file (PDF, MS Office, etc.)

-Cal uses services to implement a MVC pattern

In summary, the service architecture behaves like a factory pattern: it makes it possible to request a service without knowing exactly what class will answer the request.

## Structure of a service

- Essentially a PHP class
- Required files in an extension:
  - *ext\_localconf.php*: service registration
  - *sv1/class.tx\_myext\_sv1.php*: PHP code for service

cobweb

## Registering a service

- in `ext_localconf.php`
- using `t3lib_extMgm::addService()`

Parameters	Notes
<code>\$extKey</code>	The extension key
<code>\$serviceType</code>	The type of service
<code>\$serviceKey</code>	Unique key for the service
<code>\$info</code>	Detailed information about the service

cob.uet

### Service details

Parameters	Notes
title	Name of the service
description	Detailed description of the service
subtype	Subtypes that the service can handle
available	Whether the service is available or not (true or false)
priority	The priority of the service
quality	The quality of the service
os	The OS required for the service to work properly
exec	Any executables the service relies on
classFile	The name of the file the service is found in
className	The name of the class to call for the service

Example: various extensions provide services for the DAM

Provides many services of different subtypes.

Show view in Media > Tools > Services Info, note how some services are not available due to missing executables

Mention the purpose of priority and quality:

- services with higher priority come first
- with equal priority, higher quality comes first

## Refining a service configuration

- Override an existing configuration
  - In `typo3conf/localconf.php`
  - Syntax
    - > `$TYPO3_CONF_VARS['FE_SERVICES'][$type][$key][property] = value;`
  - Example:
    - > `$TYPO3_CONF_VARS['FE_SERVICES']['math']['tx_mx_math']['priority'] = 90;`
- Adding configuration options
  - In `typo3conf/localconf.php` or any `ext_localconf.php`
  - Syntax:
    - > `$TYPO3_CONF_VARS['@VCCONF'][$type][$key][property] = value;`
    - > `$TYPO3_CONF_VARS['@VCCONF'][$type]['defaults'][$property] = value;`
  - Retrieved using `t3lib_svbase::getServiceOption()`

Overriding configuration **MUST** be made in `typo3conf/localconf.php` since it is loaded **FIRST**. So the values defined there can influence the registration which takes places in `t3lib_extMgm::addService()`.

Adding configuration options can be made in `typo3conf/localconf.php` or in some extension's `ext_localconf.php` file. In this case the extensions load order is important of course.

## Refining a service configuration (2)

- Options outside the service
  - *Not used inside the service, but by code calling the service*
  - **Syntax:**

```
<string_conf_vars['svconf'] [type] ['setup'] [property] = value;
```
  - **Example:**

```
<string_conf_vars['svconf'] ['auth'] ['setup'] ['#_alwaysFetchOwner'] = value;
```
  - *No API for this, usage as per documentation*

cobu.net

This is really just a recommendation of syntax, a best practice.

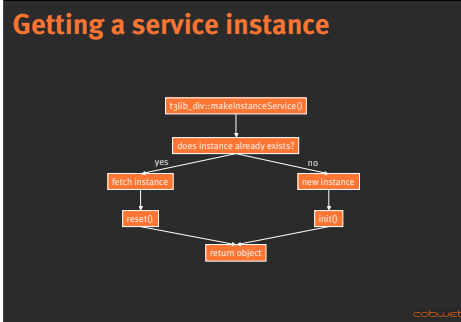
### Service-related API

- In `t3lib_extMgm`:
 

Method	Usage
<code>addService</code>	Used to add a service to the list of services. Stores all in the info in <code>\$T3_SERVICES</code> .
<code>findService</code>	Finds a specific service for a given type and subtype, taking into account priority, quality and availability.
<code>deactivateService</code>	Temporarily marks a service as unavailable.
- In `t3lib_div`:
 

Method	Usage
<code>makeInstanceService</code>	Finds an appropriate service given a type and a subtype.

Show view of `T3_SERVICES` in TYPO3 4.3



Can be as easy as calling `t3lib_div::makeInstanceService()`

But note that this keeps a registry of already instantiated classes. In effect it implements the singleton pattern.

When a new instance is created, the `init()` method is called. But when an already existing instance is returned, the `reset()` method is called, which gives the opportunity to reinitialize existing data in that instance if necessary.

Instances are stored in `$T3_VAR`, which is a badly documented global. It is mostly used to store run-time information, although in the DAM it also contains configuration information.

## Using a service chain

- Call all services of a given type/subtype

```
$exclude = '';  
while ($obj = t3lib_div::makeInstanceService($type, $subtype, $exclude)) {  
    $exclude = $obj->getServiceKey();  
    // Do something  
}
```

- Example: auth services
  - *authUser()* called inside a service chain
  - continuation of chain depends on return code

cobu.de

When calling `t3lib_div::makeInstanceService()`, it is possible to give a list of keys to ignore. By building this list inside the loop, it is possible to go through all services of a given type without calling the same service twice.

Example: the authentication process. Goes through all services (remember higher priority, higher quality goes first). When authenticating (method `authUser()`) the method will return either false, 100 or 200. If false authentication is stopped and login fails. If 100, the given service cannot authenticate, but lets the process pass on to the next service. If 200, the authentication is validated and further services need not be called. (see `t3lib/class.t3lib_userauth.php`, around line 490).

This is how you can simply override TYPO3's base authentication mechanism.

## Service API

- Base class `t3lib_svbase`
  - *Getter methods for service information*
  - *Error handling*
  - *I/O tools for reading and writing files*
  - *Service implementation*
    - > `init()`
    - > `reset()`
    - > `__destruct()`

Look at file `t3lib/class.t3lib_svbase.php`

•Getter methods: nothing special to mention, except for the `getServiceOption` method mentioned above

•Error handling: `t3lib_svbase` manages an error message stack:

- `errorPush`: puts a new error onto the stack
- `errorPull`: removes the last error in the stack (LIFO)
- `getLastError`: returns the last error number (LIFO)
- `getLastErrorMsg`: returns the last error message (LIFO) or true if no error occurred
- `getLastErrorArray`: returns the last error number and message (LIFO)
- `getErrorMsgArray`: returns the full list of error messages
- `resetErrors`: empties the error stack

•**If you implement a new type of services, you may want to use Exceptions instead.**

•I/O tools: useful when you need to read or write files. Used for example in the `metaExtract` or `textLang` services.

•Service implementation: all important methods `init()` and `reset()`, already discussed

## Implementing a service

- Existing type
  - *Inherit the base class (if any)*
  - *Implement the relevant API*
- New type
  - *Create a base class*
  - *Define own API in base class*

cobweb

Show connector services as an example.

Be careful if service is used in both BE and FE since not all globals are available in both context, e.g. the lang object (\$LANG or \$TSFE->lang).

## Looking forward

- Coming soon to a TER near you:
  - *Revamped core documentation about services (doc\_core\_services)*
- Clean up of service-related API
  - *Store instances in static variable instead of global*
- BE information module (maybe)
  - *Inspired by the DAM's info module*

cobweb