TYPO3 NEW APIS

Oliver Bartsch

@oli

@_obartsch

AGENDA

- Who’s this?
- API - What’s this?
- Facts!
- Migration / Integration?!
- Nice - What’s next?
- I have an idea / question!

WHO'S THIS?

OLIVER BARTSCH

- Working @ b13
- Core development & customer work
- Part of the TYPO3 Core Team since 2021
- TYPO3 Ecosystem & Extensions

@oli

@_obartsch

API - WHAT'S THIS?

Photo by Yassine Khalfalli on Unsplash

TYPO3

ENABLES SOFTWARE TO COMMUNICATE WITH EACH OTHER USING A SET OF DEFINITIONS AND PROTOCOLS.

API ♥ TYPO3

TYPO3

ENABLES YOU TO EXTEND AND CUSTOMIZE TYPO3 TO YOUR CUSTOM(ER) NEEDS.

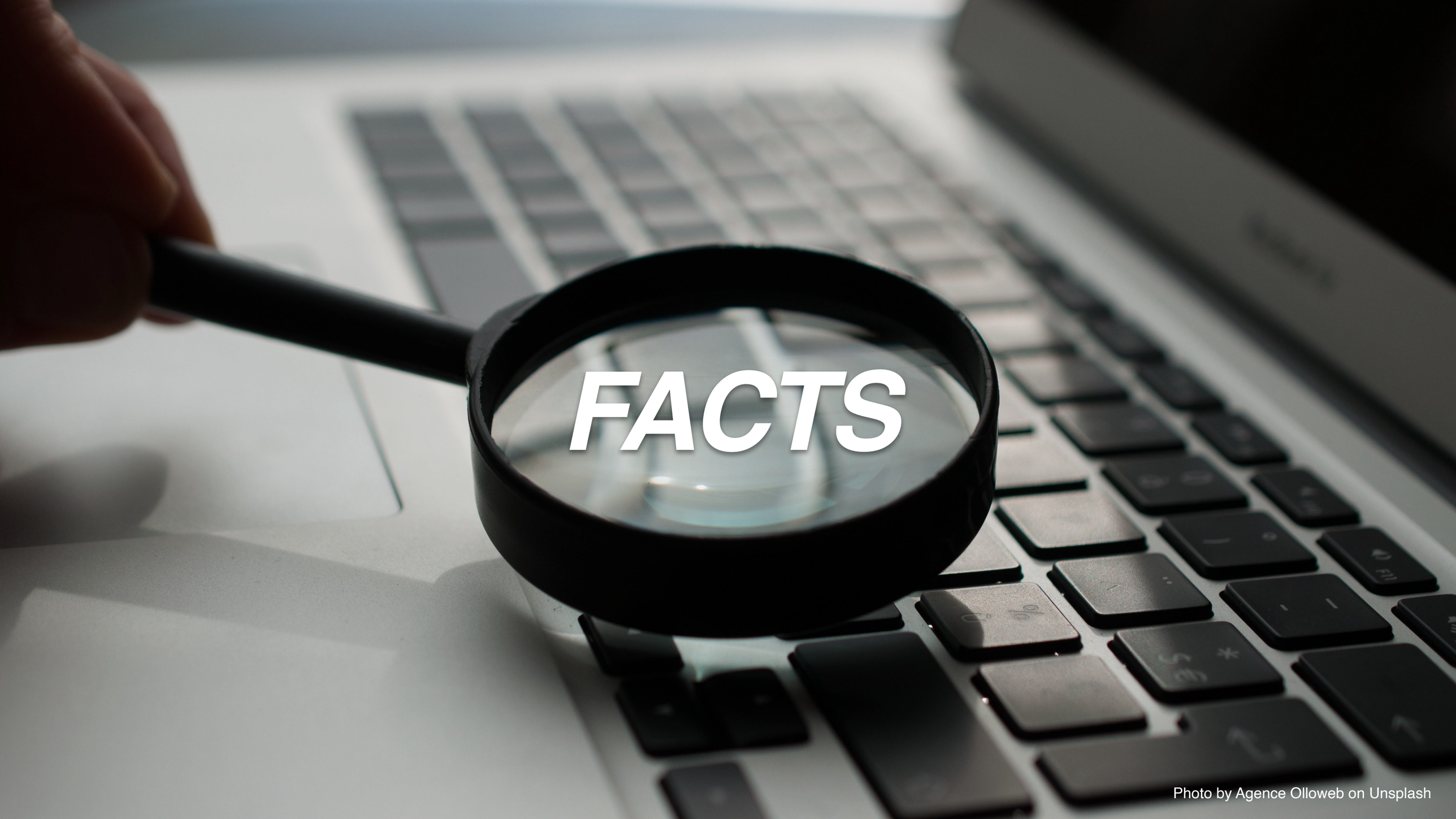


Photo by Agence Olloweb on Unsplash

TYPO3

215
FEATURES

Photo by June Gathercole on Unsplash

TYPO3

DON'T GET HOOKED
LISTEN TO EVENTS

TYPO3

> 69
PSR-14 EVENTS

ModifyPageLinkConfigurationEvent
FilterMenuItemsEvent
RecordAccessGrantedEvent
ModifyInfoModuleContentEvent
ModifyGenericBackendMessagesEvent
ModifyCacheLifetimeForPageEvent
ModifyLinkExplanationEvent
ModifyInlineElementEnabledControlsEvent
ModifyInlineElementControlsEvent
ModifyAutoCreateRedirectRecordBeforePersistingEvent
ModifyNewContentElementWizardItemsEvent
ModifyVersionDifferencesEvent
AfterCacheableContentIsGeneratedEvent
AfterCachedPageIsPersistedEvent
AfterPageTreeItemsPreparedEvent
IsContentUsedOnPageLayoutEvent
ModifyDatabaseQueryForContentEvent
PageContentPreviewRenderingEvent
BeforePageLanguageOverlayEvent
BeforeFlexFormConfigurationOverrideEvent
IsFileSelectableEvent
SiteConfigurationBeforeWriteEvent
SiteConfigurationLoadedEvent

AfterTemplatesHaveBeenDeterminedEvent
ModifyEditFormUserAccessEvent
BeforeRecordLanguageOverlayEvent
AfterRecordLanguageOverlayEvent
ModifyFileReferenceEnabledControlsEvent
EvaluateModifierFunctionEvent
ModifyQueryForLiveSearchEvent
BeforeMailerSentMessageEvent
AfterMailerSentMessageEvent
RedirectWasHitEvent
ModifyPageLayoutContentEvent
AfterLinkIsGeneratedEvent
ModifyButtonBarEvent
ShouldUseCachedPageDataIfAvailableEvent
AfterBackendPageRenderEvent
ModifyLinkHandlersEvent
BeforePagePreviewUriGeneratedEvent
AfterPagePreviewUriGeneratedEvent
BeforePageIsResolvedEvent
AfterPageWithRootLineIsResolvedEvent
AfterPageAndLanguageIsResolvedEvent
ModifyDatabaseQueryForRecordListingEvent
LoginAttemptFailedEvent

AfterRecordSummaryForLocalizationEvent
CustomFileControlsEvent
ModifyFileReferenceControlsEvent
EnrichPasswordValidationContextDataEvent
BeforeUserLogoutEvent
AfterUserLoggedOutEvent
AfterUserLoggedInEvent
AfterDefaultUploadFolderWasResolvedEvent
ModifyRedirectManagementControllerViewDataEvent
BeforeRedirectMatchDomainEvent
AfterAutoCreateRedirectHasBeenPersistedEvent
ModifyImageManipulationPreviewUrlEvent
ModifyLanguagePacksEvent
ModifyResultItemInLiveSearchEvent
AfterVideoPreviewFetchedEvent
AfterRecordPublishedEvent
ModifyBlindedConfigurationOptionsEvent
SlugRedirectChangeItemCreatedEvent
ModifyEditFileFormDataEvent
BeforeFlexFormDataStructureParsedEvent
AfterFlexFormDataStructureParsedEvent
BeforeFlexFormDataStructureIdentifierInitializedEvent
AfterFlexFormDataStructureIdentifierInitializedEvent
…

6
BACKEND MODULES

TYPO3

- Check links
- Page TSconfig
- TypoScript
- Content Security Policy
- Reactions
- Webhooks

10
TCA TYPES

TYPO3

- Color
- Datetime
- Email
- File
- Folder
- Json
- Link
- Number
- Password
- Uuid

10
TCA CONFIG OPTIONS

- Allowed
- AllowedTypes
- DbType
- Disallowed
- EnableCodeEditor
- ElementBrowserEntryPoints
- Hashed
- IgnorePageTypeRestriction
- Min
- Nullable
- PasswordPolicy
- Required

TYPO3

PHP
ATTRIBUTES

- *Backend controller*
- *Extbase annotations*
- *Webhook messages*
- *Upgrade wizards*

NEW AND IMPROVED REGISTRATION

- Upgrade wizards
- Webhooks
- Reactions
- Data processors
- Extbase type converters
- Content objects
- Element browsers
- Reports and Statuses
- Module functions
- Backend toolbar items
- Context menu item providers
- Linkvalidator linktypes

Photo by Holly Mandarich on Unsplash

Photo by Nathan Dumlao on Unsplash

TYPO3

HOOKS → EVENTS

- Consistent **registration**
- Unified **structure**
- Simplified **usage**
- Easily **extendable**
- Out-of-the-box support for **webhooks**

```php
$GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']['t3lib/class.t3lib_userauth.php']
['postLoginFailureProcessing']['sendEmailOnFailedLoginAttempt'] =
FailedLoginAttemptNotification::class . '->sendEmailOnLoginFailures';
```

```yaml
TYPO3\CMS\Backend\EventListener\FailedLoginAttemptNotification:
    tags:
      - name: event.listener
        identifier: 'typo3/cms-backend/failed-login-attempt-notification'
        event: TYPO3\CMS\Core\Authentication\Event\LoginAttemptFailedEvent
        method: 'sendEmailOnLoginFailures'
```

```yaml
TYPO3\CMS\Backend\EventListener\FailedLoginAttemptNotification:
    tags:
      - name: event.listener
        identifier: 'typo3/cms-backend/failed-login-attempt-notification'
      - name: event.listener
        identifier: 'typo3/cms-backend/failed-mfa-verification-notification'
```

```php
public function __invoke(
  LoginAttemptFailedEvent|MfaVerificationFailedEvent $event
): void {
}
```

```php
class WizardItemsHook implements NewContentElementWizardHookInterface
{
    public function manipulateWizardItems(
        &$wizardItems,
        &$parentObject
    ): void {
    }
}
```

```php
class WizardItemsListener
{
    public function manipulateWizardItems(
        ModifyNewContentElementWizardItemsEvent $event
    ): void {
        $event->hasWizardItem('itemIdentifier');
        $event->getWizardItem('itemIdentifier');
        $event->getWizardItems();
        $event->setWizardItem('newItem', [], ['after' => 'itemIdentifier']);
        $event->setWizardItems([]);
    }
}
```

Photo by israel palacio on Unsplash

- Based on PHP (implementation) classes & database records
- TYPO3 provides built-in reactions and webhooks
- Management via dedicated backend modules
- Fully customizable and extensible

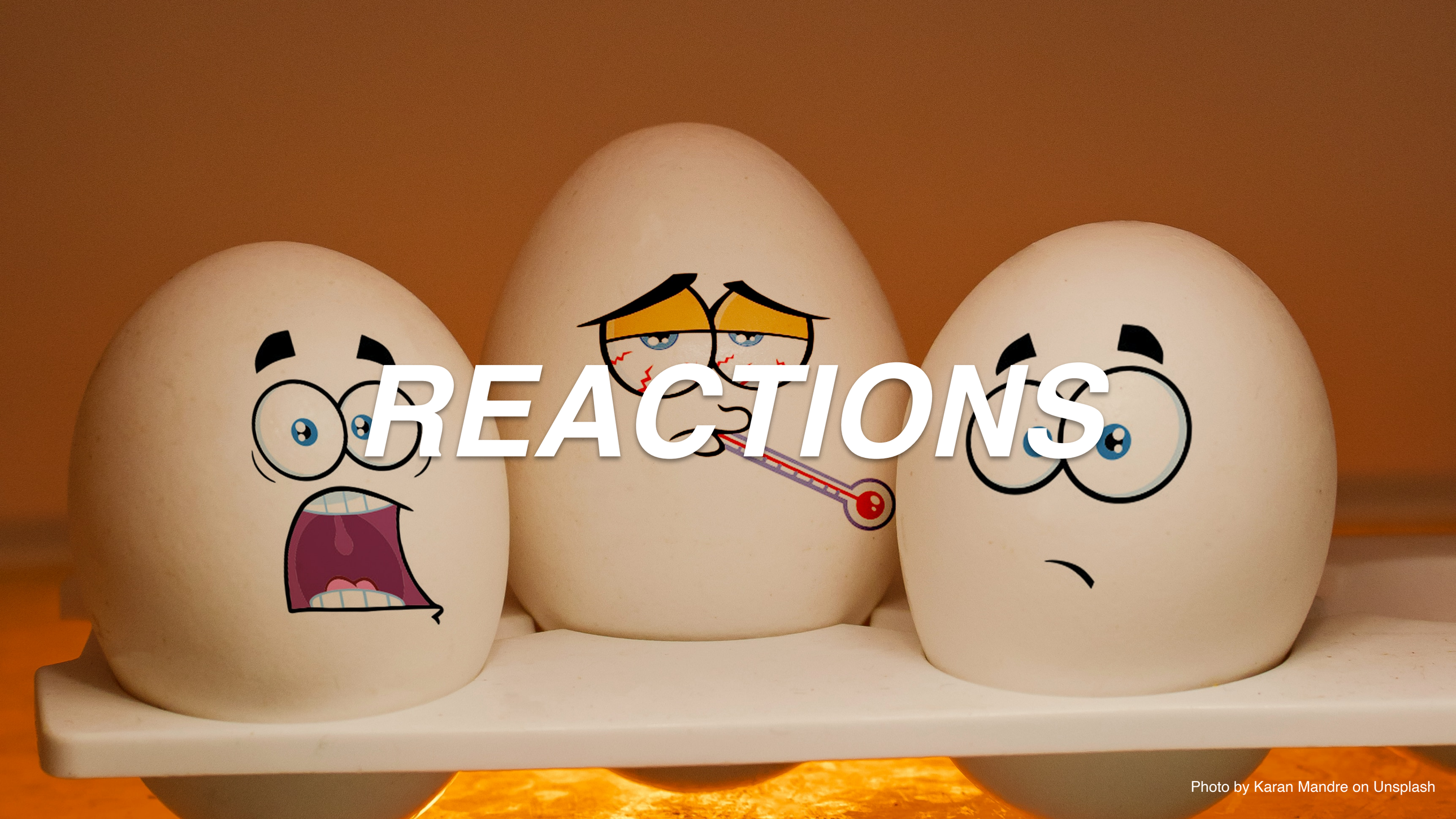
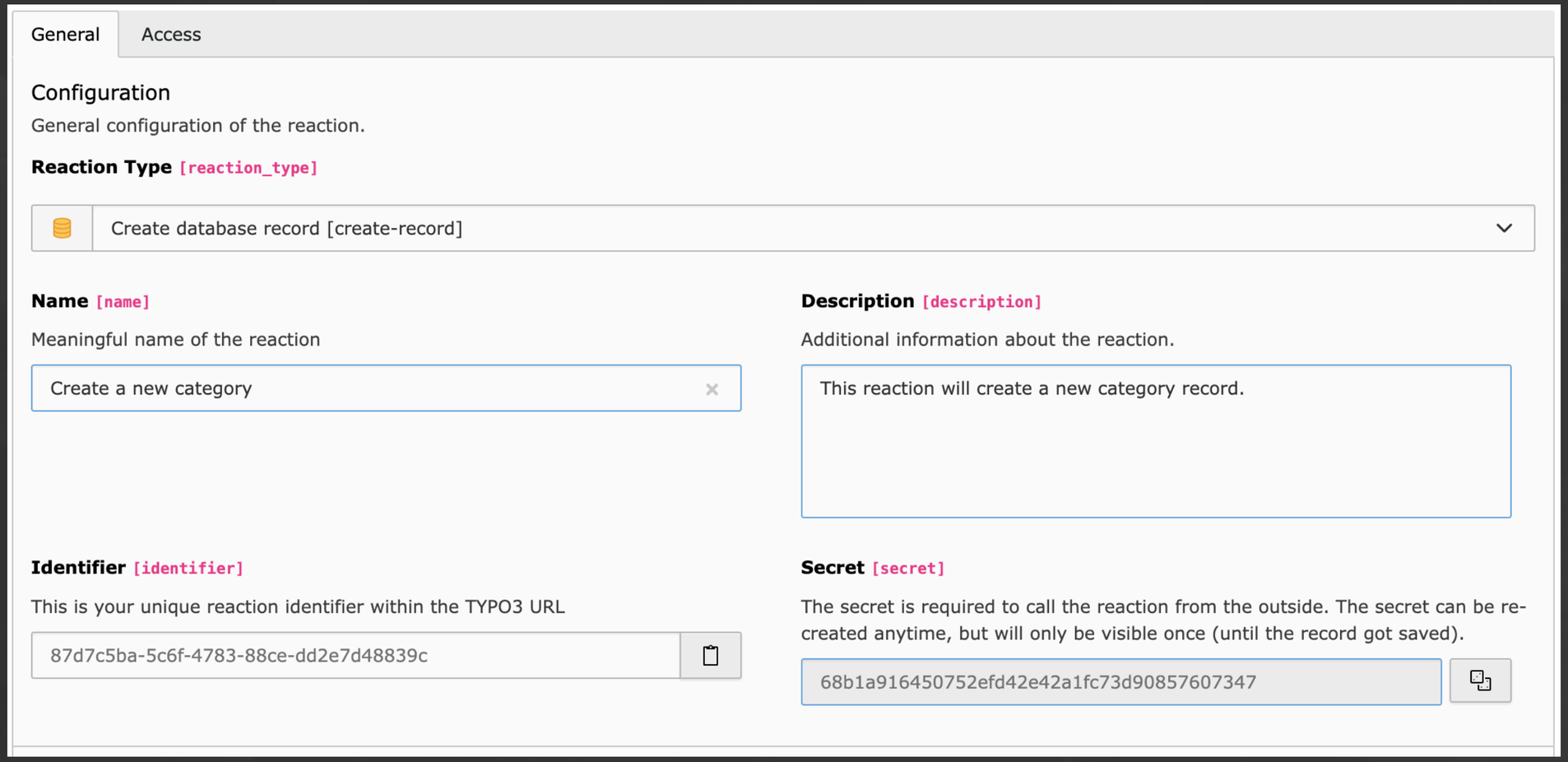
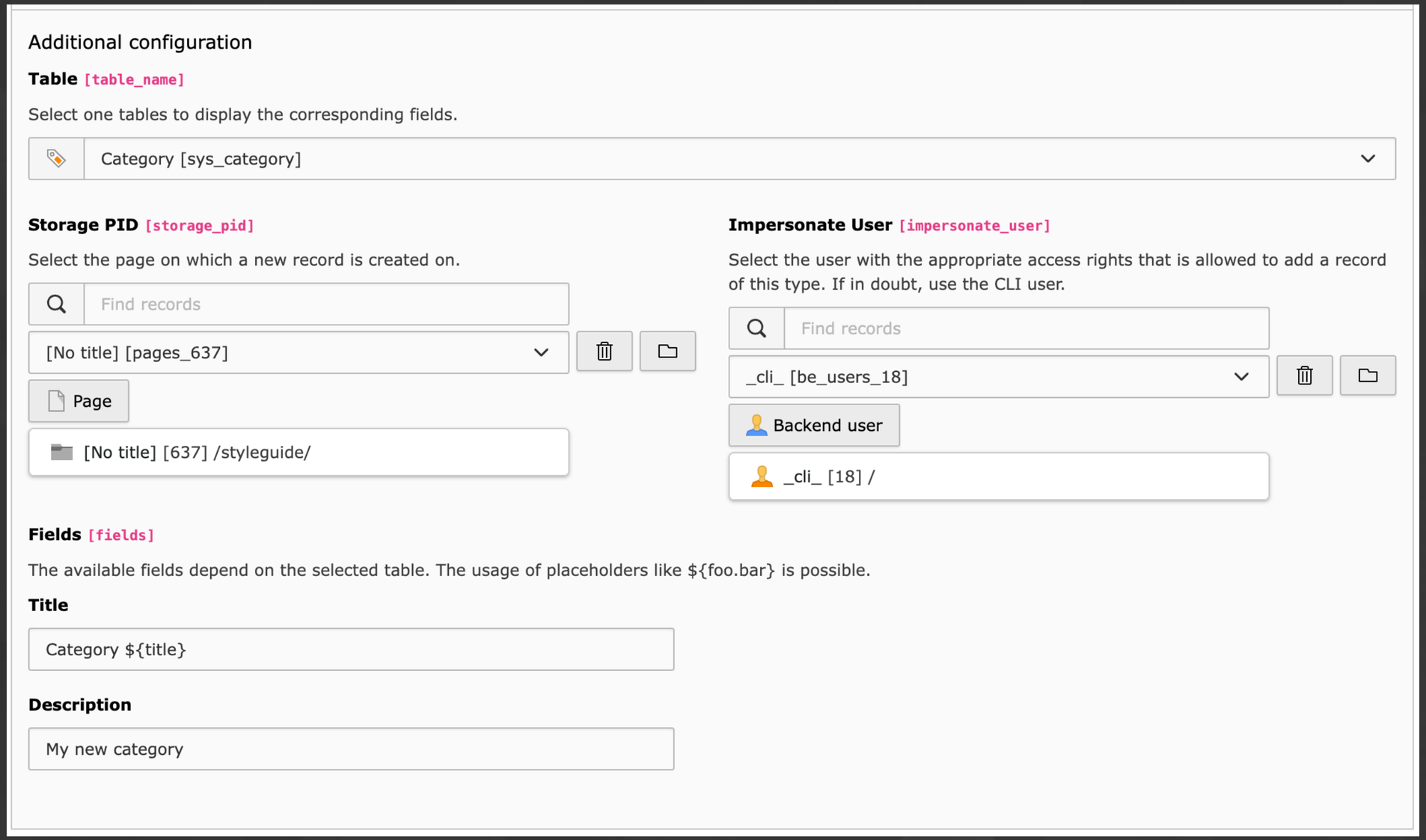


Photo by Karan Mandre on Unsplash


General
Access
Configuration
General configuration of the reaction.
Reaction Type [reaction_type]
Create database record [create-record]
Name [name]
Meaningful name of the reaction
Create a new category
Description [description]
Additional information about the reaction.
This reaction will create a new category record.
Identifier [identifier]
This is your unique reaction identifier within the TYPO3 URL
87d7c5ba-5c6f-4783-88ce-dd2e7d48839c
Secret [secret]
The secret is required to call the reaction from the outside. The secret can be re-created anytime, but will only be visible once (until the record got saved).
68b1a916450752efd42e42a1fc73d90857607347


Additional configuration
Table [table_name]
Select one tables to display the corresponding fields.
Category [sys_category]
Storage PID [storage_pid]
Select the page on which a new record is created on.
Find records
[No title] [pages_637]
Page
[No title] [637] /styleguide/
Impersonate User [impersonate_user]
Select the user with the appropriate access rights that is allowed to add a record of this type. If in doubt, use the CLI user.
Find records
cli [be_users_18]
Backend user
cli [18] /
Fields [fields]
The available fields depend on the selected table. The usage of placeholders like ${foo.bar} is possible.
Title
Category ${title}
Description
My new category

```
curl -X 'POST' \
    'https://typo3.site/typo3/reaction/87d7c5ba-5c6f-4783-88ce-dd2e7d48839c' \
      -H 'accept: application/json' \
      -H 'x-api-key: 68b1a916450752efd42e42a1fc73d90857607347' \
      -H 'Content-Type: application/json' \
      -d '{"title":"dynamic title"}
```

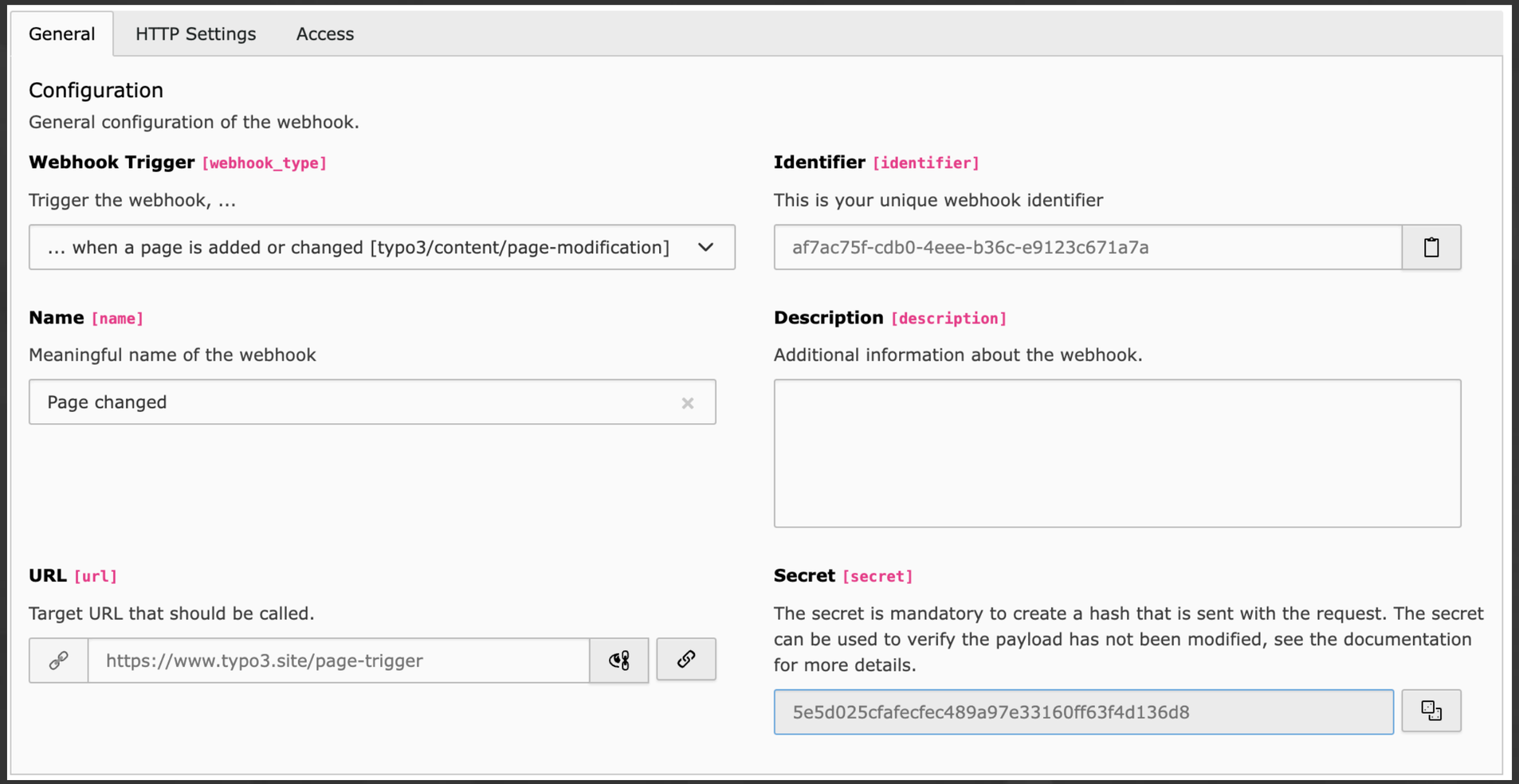
```
{
    "success": true
}
```

- Registration via **ReactionInterface**
- Handling in **react()** method
- TCA registration and configuration

```php
final class MyReaction implements ReactionInterface
{
    public static function getType(): string
    {
        return 'my-reaction';
    }

    public static function getDescription(): string
    {
        return 'My simple reaction';
    }

    public static function getIconIdentifier(): string
    {
        return 'actions-info';
    }

    public function react(
      ServerRequestInterface $request,
      array $payload,
      ReactionInstruction $reaction
    ): ResponseInterface {
        return new JsonResponse(['success' => true]);
    }
}
```

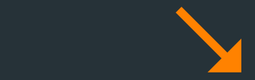
Photo by Icons8 Team on Unsplash

- **PHP class, a**uto-tagged using **WebhookMessage** Attribute
  - identifier
  - description
  - (factory) method - optional
- Auto-configuration via CompilerPass
- Automatic TCA registration

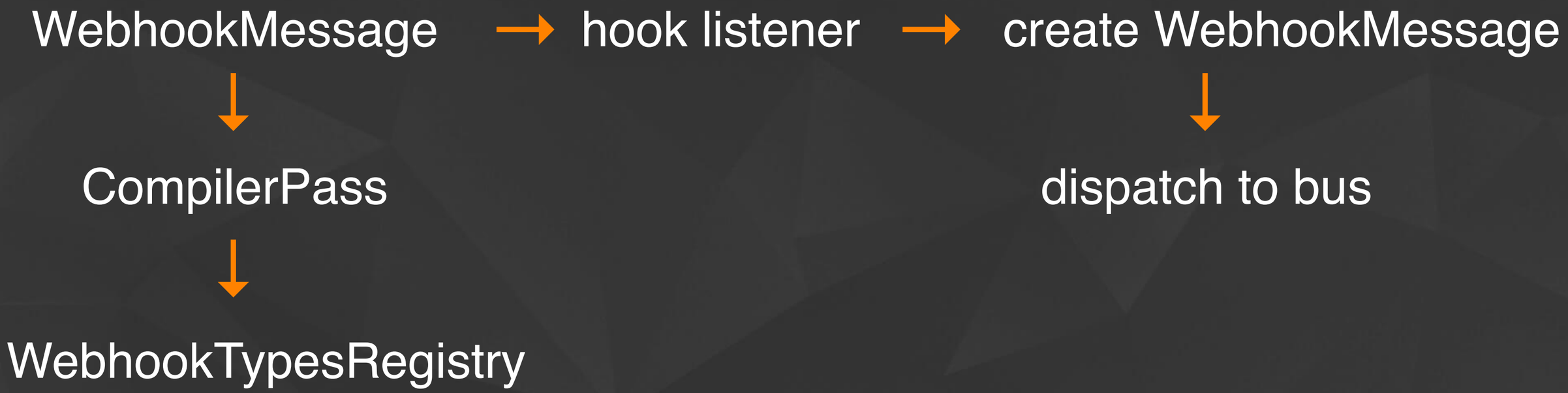

General
HTTP Settings
Access
Configuration
General configuration of the webhook.
Webhook Trigger [webhook_type]
Trigger the webhook, ...
... when a page is added or changed [typo3/content/page-modification]
Identifier [identifier]
This is your unique webhook identifier
af7ac75f-cdb0-4eee-b36c-e9123c671a7a
Name [name]
Meaningful name of the webhook
Page changed
Description [description]
Additional information about the webhook.
URL [url]
Target URL that should be called.
https://www.typo3.site/page-trigger
Secret [secret]
The secret is mandatory to create a hash that is sent with the request. The secret can be used to verify the payload has not been modified, see the documentation for more details.
5e5d025cfafecfec489a97e33160ff63f4d136d8


General
HTTP Settings
Access
HTTP Settings
Advanced HTTP request settings.
HTTP Method [method]
POST [POST]
Verify SSL [verify_ssl]
Verify that the connection is secure and valid, this setting should only be disabled in case a verification is not possible.
[1]
Additional Headers [additional_headers]
Additional headers that should be added to the HTTP request. Data must be provided as valid JSON string.
{
"X-CUSTOM-HEADER": "New API"
}

```
X-Custom-Header: New API
Webhook-Signature-Algo: sha256
Webhook-Signature: 642934714fa90cc0215bde2a9e1f8080dee096db12ea923c36ee4442a54147da
Content-Type: application/json
Content-Length: 1311
User-Agent: TYPO3
Host: typo3.site

{
  "action": "new",
  "identifier": 1186,
  "record": {
    "title": "new-page",
    "slug": "/webhooks/new-page",
  },
  "url": "https://typo3.site/webhooks/new-page.html",
  "site": "styleguide",
  "workspace": 0,
  "author": {
    "uid": 1,
    "username": "oli",
    "isAdmin": true
  }
}
```



```php
hash_hmac('sha256', sprintf(
    '%s:%s',
    $webhookRecordIdentifier,
    $requestBody,
), $webhookRecordSecret);
```


WebhookMessage
hook listener
create WebhookMessage
CompilerPass
dispatch to bus
WebhookTypesRegistry

WebhookMessageHandler

↘ *WebhookTypesRegistry*

getConfiguredWebhooksByType($message)

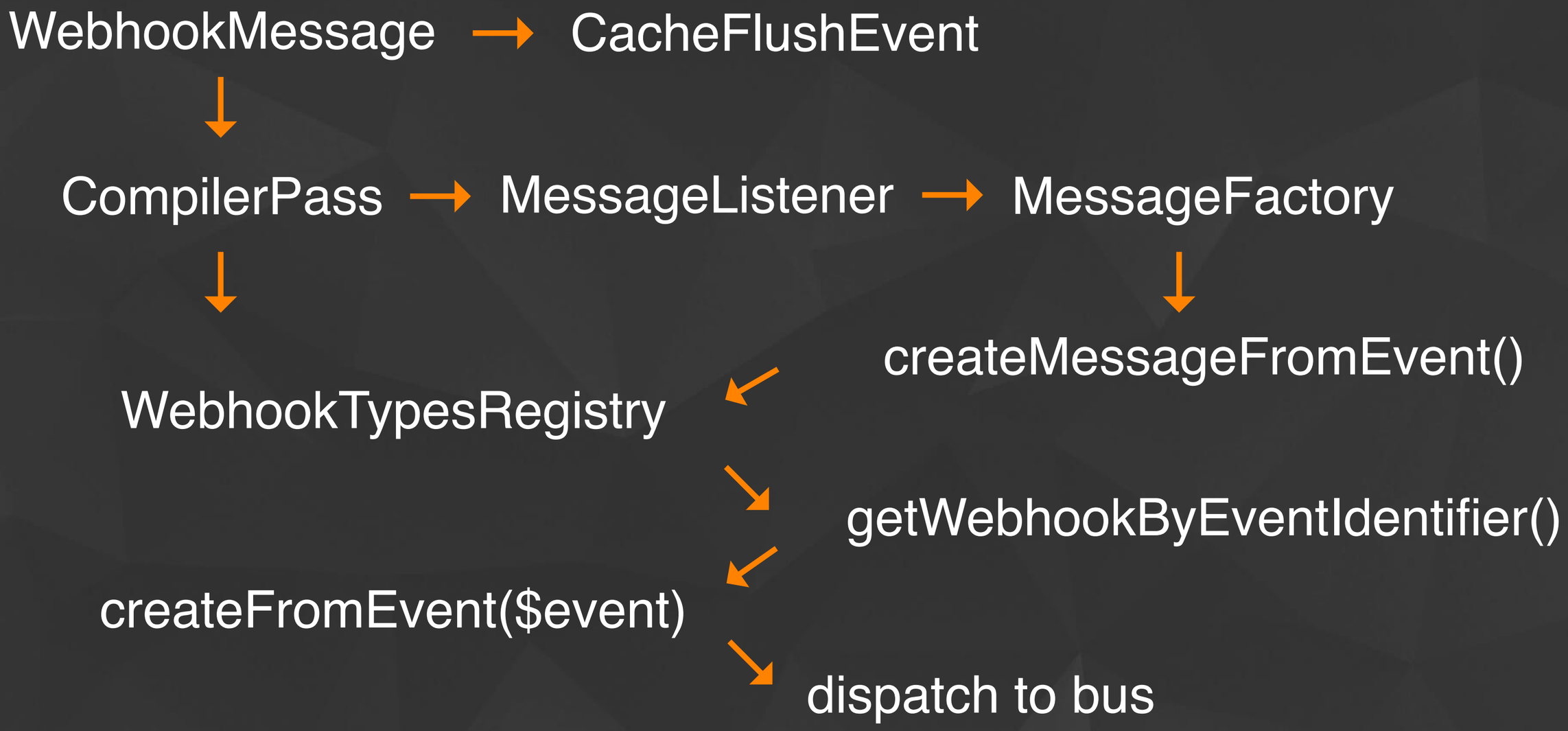
↘ *WebhookInstructions*

sendRequest($webhookInstruction, $message)

USING EVENTS

```php
#[WebhookMessage(
    identifier: 'my-ext/cache-flushed',
    description: '... when cache is flushed'
)]
final class CacheFlushMessage implements WebhookMessageInterface
{
    public function __construct(
        private readonly array $errors,
        private readonly array $groups
    ) {
    }

    public static function createFromEvent(CacheFlushEvent $event): self
    {

        return new self($event->getErrors(), $event->getGroups());
    }

    public function jsonSerialize(): array
    {
        return [
            'errors' => $this->errors,
            'groups' => $this->groups,
        ];
    }
}
```

- Handling via factory method **createFromEvent($event)**
- **MessageListener** listening for $event
- **MessageFactory** calling factory method


WebhookMessage
CacheFlushEvent
CompilerPass
MessageListener
MessageFactory
createMessageFromEvent()
WebhookTypesRegistry
getWebhookByEventIdentifier()
createFromEvent($event)
dispatch to bus

TYPO3

ISN'T THIS EASY?

TCA

TYPES & OPTIONS

TYPO3

LESS CONFIGURATION
MORE CLARITY

```php
'file' => [
    'label' => 'File field',
    'config' => [
        'type' => 'inline',
        'foreign_table' => 'sys_file_reference',
        'foreign_field' => 'uid_foreign',
        'foreign_sortby' => 'sorting_foreign',
        'foreign_table_field' => 'tablenames',
        'foreign_match_fields' => [
            'fieldname' => 'file',
        ],
        'foreign_label' => 'uid_local',
        'foreign_selector' => 'uid_local',
        'overrideChildTca' => [
            'columns' => [
                'uid_local' => [
                    'config' => [
                        'appearance' => [
                            'elementBrowserType' => 'file',
                            'elementBrowserAllowed' => 'jpg,png',
                        ],
                    ],
                ],
            ],
        ],
        'filter' => [
            [
                'userFunc' => FileExtensionFilter::class . '->filterInlineChildren',
                'parameters' => [
                    'allowedFileExtensions' => $GLOBALS['TYPO3_CONF_VARS']['GFX']['imagefile_ext'],
                    'disallowedFileExtensions' => 'gif',
                ],
            ],
        ],
    ],
],
```

```php
'file' => [
    'label' => 'File field',
    'config' => [
        'type' => 'file',
        'allowed' => ['common-media-types'],
        'disallowed' => ['gif']
    ],
],
```

```
'password_field' => [
    'label' => 'Password',
    'config' => [
        'type' => 'input',
        'eval' => 'trim,password,saltedPassword',
    ]
],
```

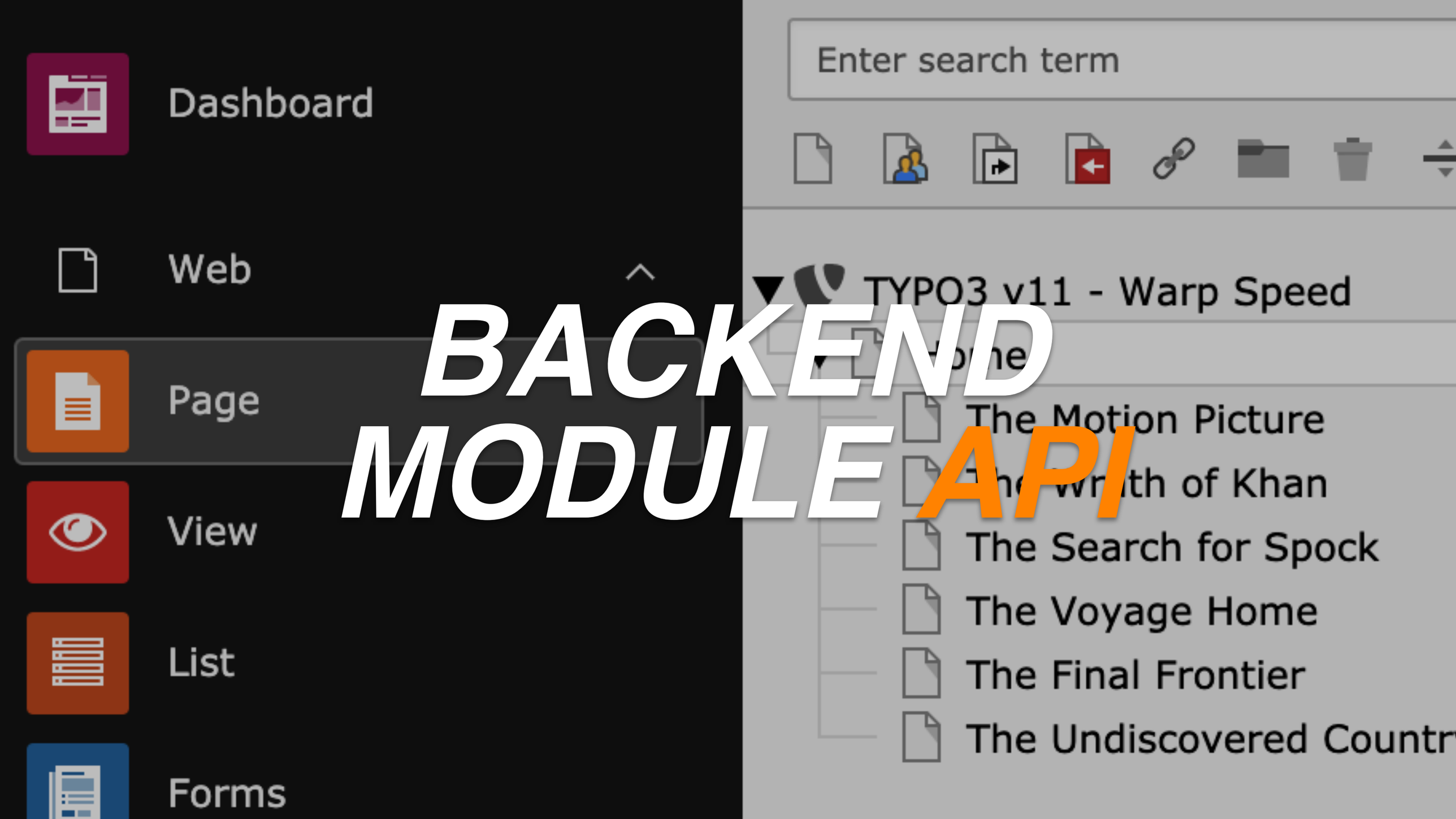
```
'password_field' => [
    'label' => 'Password',
    'config' => [
        'type' => 'password',
        'passwordPolicy' => 'custom',
        'fieldControl' => [
            'passwordGenerator' => [
                'renderType' => 'passwordGenerator',
                'options' => [
                    'title' => 'Generate a password',
                    'allowEdit' => false,
                    'passwordRules' => [
                        'length' => 38,
                        'digitCharacters' => false,
                        'specialCharacters' => true,
                    ],
                ],
            ],
        ],
    ],
],
```

Photo by Paulius Dragunas on Unsplash

- Enforce password policies in backend and frontend
- Built-in policies and validators
- Extendable and customizable
- Dedicated TCA type with options and field control
- PSR-14 Event to enrich validation context data

CONTENT SECURITY POLICY

Photo by Scott Webb on Unsplash



BACKEND MODULE API

- *$TBE_MODUELS* and *ModuleLoader* **removed**
- Registration during **build-time** in *ModuleRegistry*
- Configuration in *Configuration/Backend/Modules.php*
- Access modules via ***ModuleProvider API***
- Module object in PSR-7 Request
- *PSR-14 Event for manipulating module config*

```php
ExtensionManagementUtility::addModule(
    'web',
    'module',
    'top',
    '',
    [
        'routeTarget' => MyModuleController::class . '::handleRequest',
        'name' => 'web_module',
        'access' => 'admin',
        'workspaces' => 'online',
        'iconIdentifier' => 'module-example',
        'labels' => 'LLL:EXT:example/Resources/Private/Language/locallang_mod.xlf',
        'navigationComponentId' => 'TYPO3/CMS/Backend/PageTree/PageTreeElement',
    ]
);
```

TYPO3

```php
return [
    'web_module' => [
        'parent' => 'web',
        'position' => ['before' => '*'],
        'access' => 'admin',
        'workspaces' => 'live',
        'path' => '/module/web/example',
        'iconIdentifier' => 'module-example',
        'navigationComponent' => 'TYPO3/CMS/Backend/PageTree/PageTreeElement',
        'labels' => 'LLL:EXT:example/Resources/Private/Language/locallang_mod.xlf',
        'routes' => [
            '_default' => [
                'target' => MyExampleModuleController::class . '::handleRequest',
            ],
        ],
    ],
];
```

```
'extensionName' => 'MyExtbaseExtension',
'controllerActions' => [
    MyExtbaseExtensionModuleController::class => [
        'index',
        'detail'
    ],
],
```

SPECIFIC ROUTES & DYNAMIC URL PARTS

```php
return [
    'web_module' => [
        'parent' => 'web',
        'path' => '/module/web/example',
        'routes' => [
            '_default' => [
                'target' => MyModuleController::class . '::overview',
            ],
            'edit' => [
                'path' => '/custom-path',
                'target' => MyModuleController::class . '::edit',
            ],
            'manage' => [
                'target' => AnotherController::class . '::manage',
                'methods' => ['POST'],
            ],
            'delete' => [
                'path' => '/delete/{identifier}',
                'target' => AnotherController::class . '::delete',
                'methods' => ['POST'],
            ],
        ],
    ],
];
```

```php
public function delete(ServerRequestInterface $request): ResponseInterface
{
    $identifier = $request->getAttribute('routing')['identifier'];
}
```

ALIASES

TYPO3

```php
return [
    'new_name' => [
        'parent' => 'web',
        ...
        'aliases' => ['web_module'],
    ],
];
```

MODULE FUNCTIONS API

- *$TBE_MODUELS_EXT* **removed**
- Definition of access permissions possible
- Registration and functionality same as for “main” modules

```php
ExtensionManagementUtility::insertModuleFunction(
    'new_name',
    MyThirdLevelController::class,
    '',
    'LLL:EXT:extkey/Resources/Private/Language/locallang.xlf:mod_title'
);
```

```
'new_name_third' => [
    'parent' => 'new_name',
    'path' => '/module/new/name/thrid',
    'labels' => [
        'title' => 'LLL:EXT:extkey/Resources/Private/Language/locallang.xlf:mod_title',
    ],
    'routes' => [
        '_default' => [
            'target' => MyThirdLevelController::class . '::handleRequest',
        ],
    ],
    'moduleData' => [
        'someData' => 0,
    ],
],
```

MODULE DATA API

- Stores user related module state
- Definition of allowed GET/POST parameters
- ModuleData object available in PSR-7 Request

```php
return [
    'new_name' => [
        'parent' => 'web',
        ...
        'moduleData' => [
            'language' => 0,
            'showHiddenElements' => true,
        ],
    ],
];
```

TYPO3

```php
$moduleData = $request->getAttribute('moduleData');
$moduleIdentifier = $request->getAttribute('module')->getIdentifier();

if ($moduleData->clean('showHiddenElements', [true, false])) {
    $backendUser->pushModuleData($moduleIdentifier, $moduleData->toArray());
}

$showHiddenElements = $moduleData->get('showHiddenElements');
```

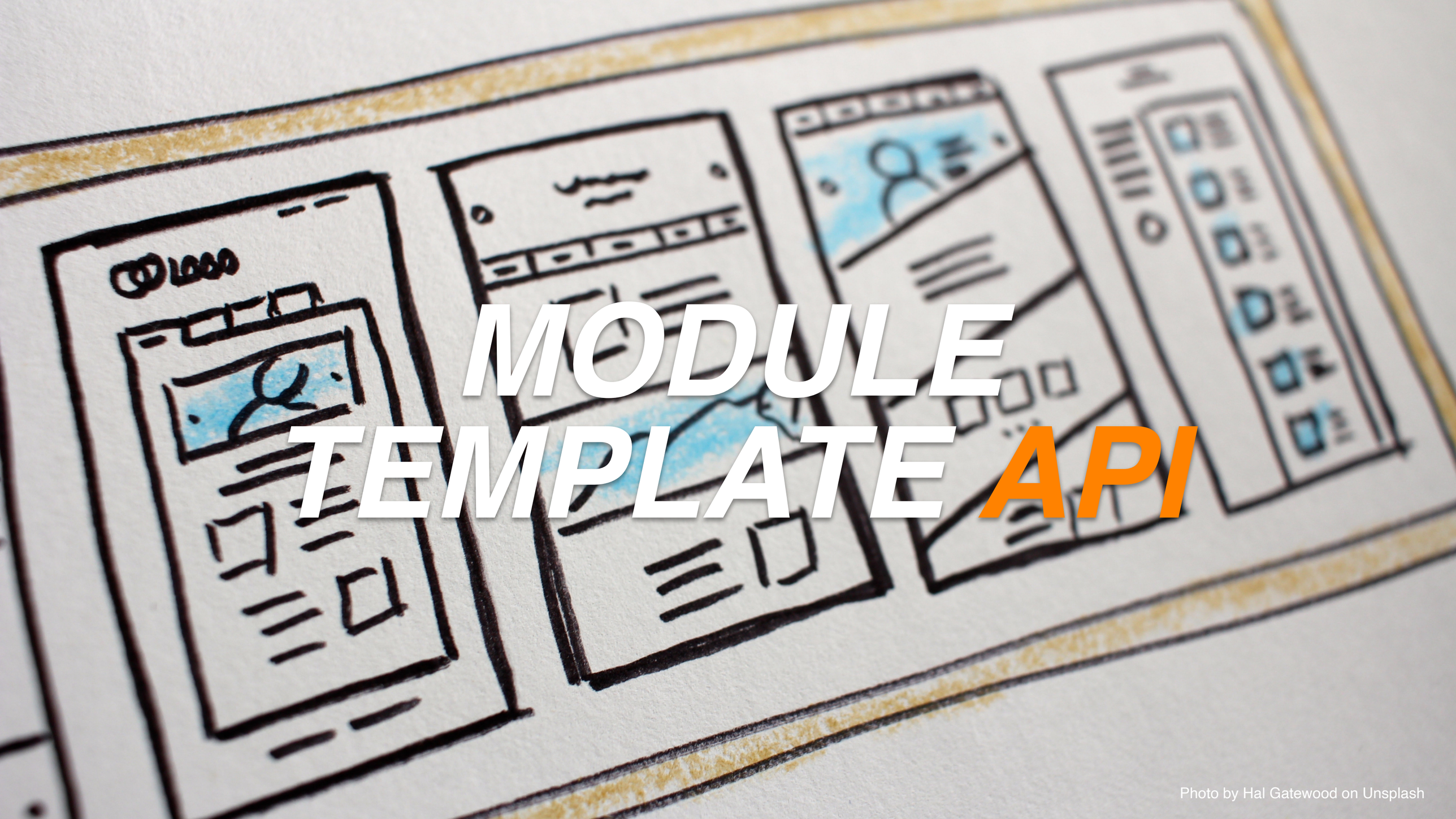


Photo by Hal Gatewood on Unsplash

- Rendering of the backend module content frame
- Auto-discover of template paths
- Auto-creation of the PSR-7 Response
- Support of global template overrides

```php
$moduleTemplate = $this->moduleTemplateFactory->create($request);
$view = GeneralUtility::makeInstance(StandaloneView::class);
$view->setTemplateRootPaths(['EXT:my_ext/Resources/Private/Templates']);
$view->assign('content', 'My module content');
$moduleTemplate->setContent($view->render('MyModuleTemplate'));
return $this->responseFactory->createResponse()
    ->withHeader('Content-Type', 'text/html; charset=utf-8')
    ->withBody($this->streamFactory->createStream($moduleTemplate->renderContent()));
```

```php
return $this->moduleTemplateFactory
    ->create($request)
    ->assign('content', 'My module content')
    ->renderResponse('MyModuleTemplate');
```

Photo by Edge2Edge Media on Unsplash

- New **LinkFactory** based on **LinkResult(Interface)**
- Enhanced **LinkResult**
  - Provides standard **output formats** out-of-the-box
- PSR-14 Event to modify **any** generated link

```php
$GLOBALS['TSFE']->cObj->typoLink($linkText, ['parameter' => $linkHref]);
$linkResult = $GLOBALS['TSFE']->cObj->lastTypoLinkResult;
if ($linkResult) {
    $linkHref = $linkResult->getUrl();
    $linkText = $linkResult->getLinkText();
    $attributes = $linkResult->getAttributes();
}
```

```php
$linkFactory = GeneralUtility::makeInstance(LinkFactory::class);
try {
    $linkResult = $linkFactory->create($linkText, ['parameter' => $linkHref], $GLOBALS['TSFE']->cObj);
    $linkText = $linkResult->getLinkText();
    $attributes = $linkResult->getAttributes();
} catch (UnableToLinkException $e) {
}
```

TYPO3

```php
$linkResult = $GLOBALS['TSFE']->cObj->typoLink($linkText, ['returnLast' => 'result']);
$linkResult = $GLOBALS['TSFE']->cObj->createLink($linkText, $conf);
$linkResult = $linkFactory->create($linkText, $conf, $GLOBALS['TSFE']->cObj);

$html = LinkResult::adapt($linkResult, LinkResult::::STRING_CAST_HTML);
$html = $linkResult->withFlags(LinkResult::::STRING_CAST_HTML);

$json = LinkResult::adapt($linkResult, LinkResult::STRING_CAST_JSON);
$json = $linkResult->withFlags(LinkResult::STRING_CAST_JSON);

return (string)$html
return $linkResult->getHtml();

return (string)$json;
return $linkResult->getJson();
return json_encode($linkResult->toArray())
return json_encode($linkResult);
```

Photo by Rima Kruciene on Unsplash

- **SiteSettings** (READ) API
- Dedicated **settings file**
- TypoScript **getData** support
- PSR-14 Events to manipulate settings

- Glob pattern in imports
- Site config available in configuration module

TYPO3

```php
$siteSettings = $request->getAttribute('site')->getSettings();

$allSettings = $siteSettings->all();
$allSitePids = $siteSettings->get('sites');
$newsDetailPid = (int)$siteSettings->get('sites.newsDetail', 110);
```

Photo by S O C I A L . C U T on Unsplash

- Replace last Hooks with PSR-14 Events
- Add even **more** PSR-14 Events
- Improve TCA API
- Improve Settings API
- Doctrine DBAL 4 & CommonTableExpressions API

QUESTIONS?

IDEAS?

TYPO3

THANK YOU!