Migrations-Guide: SqueezeConfig zu TenantConfig

In Squeeze Version 2.32 wird die seit 2021 veraltete und als "deprecated" markierte Klasse `SqueezeConfig` endgültig durch den neuen, über Dependency Injection (DI) verfügbaren Service `TenantConfig` ersetzt. Dieser Guide beschreibt, wie bestehende UserExits, die `SqueezeConfig` verwenden, auf den neuen Standard migriert werden können.

Hintergrund

`SqueezeConfig` basierte oft auf statischen Zuständen oder manueller Instanziierung, was die Testbarkeit und Flexibilität einschränkte. Der neue `TenantConfig`-Service ist Teil der V2-Architektur und nutzt den modernen DI-Container von SQUEEZE.

Was hat sich geändert?

Merkmal	Alt: `SqueezeConfig`	Neu: `TenantConfig`
Namespace	`Squeeze\SqueezeConfig`	`App\V2\Base\Config\TenantConfig`
Instanziierung	`new SqueezeConfig()`	Über Dependency Injection oder den Service-Container
Konfigurationszugriff	`$config->get('key')`	`$config->get('key')` (wenn der Wert `null` sein darf) oder `$config->mustGet('key')` (wenn der Wert immer gesetzt sein muss)
Typsicherheit	Rückgabe meist `string` oder `mixed`	Spezifische Methoden wie `getInt()`, `getBool()`, `getArray()`

Migration von UserExits

In UserExits gibt es zwei primäre Wege, um auf die Konfiguration zuzugreifen.

> **Hinweise**

> **Namespaces:** Achten Sie beim Import von `TenantConfig` auf den korrekten Namespace `App\V2\Base\Config\TenantConfig`.
> **Statische Methoden:** Vermeiden Sie statische Zugriffe auf Konfigurationen, wo immer möglich. Nutzen Sie die Instanz-Methoden des `TenantConfig`-Services.

1. In klassischen Funktions-basierten UserExits (Legacy)

Wenn Ihr UserExit eine einfache PHP-Funktion ist (z. B. `AfterExtraction`), die eine `xDoc`-Instanz erhält, aber keinen Zugriff auf DI hat, können Sie den Service-Container manuell abfragen:

Vorher (SqueezeConfig):

```
use Squeeze\SqueezeConfig;

function AfterExtraction(xDoc $xDoc) {
    $config = new SqueezeConfig();
    $repo = $config->get('repository.root');
    // ...
}
```

Nachher (TenantConfig über Container):

```
use App\V2\Base\Config\TenantConfig;
use App\V2\Base\DI\Dependencies;

function AfterExtraction(xDoc $xDoc) {
    /** @var TenantConfig $config */
    $config = Dependencies::getContainer()->get(TenantConfig::class);

    // Zugriff mit Standardwert
    $repo = $config->get('repository.root');

    // Zugriff mit Exception, falls Key fehlt (sicherer)
    $repo = $config->mustGet('repository.root');

    // Typsicherer Zugriff
    $maxSize = $config->getInt('upload.maxSize', 1024);
    // ...
}
```

2. In modernen V2 UserExits (Klassen-basiert)

Wenn Sie bereits V2-UserExits verwenden, die als Klassen implementiert sind (z. B. Erben von `AbstractUserExitBase`), sollten Sie `TenantConfig` einfach per Constructor Injection anfordern.

Beispiel:

```
namespace App\V2\Modules\UserExit\UserExits;

use App\V2\Base\Config\TenantConfig;

class MyCustomUserExit extends AbstractUserExitBase {
    public function __construct(
        protected UserExitRegistry $userExitRegistry,
        private TenantConfig $tenantConfig // Neu hinzugefügt
    ) {
        parent::__construct($userExitRegistry);
    }

    public function call(xDoc $xDoc) {
        $repo = $this->tenantConfig->mustGet('repository.root');
        // ...
    }
}
```

Wichtige Methoden im Vergleich

SqueezeConfig	TenantConfig (Empfehlung)	Beschreibung
`get($key, $default)`	`get(string $key, $default = null)`	Gibt den Wert oder den Default zurück.
-	`mustGet(string $key)`	Wirft eine `MissingConfigException`, wenn der Key fehlt.
`getElasticsearchIndex()`	`get('elasticSearch.index')`	Der Index wird nun direkt über den Key abgefragt.
`getClientName()`	`get('tenant.name')` (oder über `TenantContext`)	Zugriff auf den Mandantennamen.

Revision #9
Created 2026-03-25 16:46:07 UTC by Florian Vick
Updated 2026-04-13 12:51:34 UTC by Florian Vick