Contao SensorFetcher Erweiterungs-Bundle Beispiel

**Dieses Dokument beschreibt, wie du ein separates Contao-Bundle erstellst, das eigene SensorFetcher für dein bestehendes ContaoHab-System registriert. Das Bundle bleibt modular und unabhängig vom Hauptsystem.**

# Verzeichnisstruktur

src/Acme/AcmeSensorExtensionBundle/  
├── AcmeSensorExtensionBundle.php  
├── DependencyInjection/  
│ └── AcmeSensorExtensionExtension.php  
├── Sensor/  
│ └── DummySensorFetcher.php  
├── Resources/  
│ └── config/  
│ └── services.yaml  
├── composer.json

# 1. AcmeSensorExtensionBundle.php

<?php  
  
namespace Acme\AcmeSensorExtensionBundle;  
  
use Symfony\Component\HttpKernel\Bundle\Bundle;  
  
class AcmeSensorExtensionBundle extends Bundle  
{  
}

# 2. DependencyInjection/AcmeSensorExtensionExtension.php

<?php  
  
namespace Acme\AcmeSensorExtensionBundle\DependencyInjection;  
  
use Symfony\Component\DependencyInjection\ContainerBuilder;  
use Symfony\Component\DependencyInjection\Extension\Extension;  
use Symfony\Component\DependencyInjection\Loader\YamlFileLoader;  
use Symfony\Component\Config\FileLocator;  
  
class AcmeSensorExtensionExtension extends Extension  
{  
 public function load(array $configs, ContainerBuilder $container): void  
 {  
 $loader = new YamlFileLoader($container, new FileLocator(\_\_DIR\_\_ . '/../Resources/config'));  
 $loader->load('services.yaml');  
 }  
}

# 3. Sensor/DummySensorFetcher.php

<?php  
  
namespace Acme\AcmeSensorExtensionBundle\Sensor;  
  
use PbdKn\ContaoContaohabBundle\Model\SensorModel;  
use PbdKn\ContaoContaohabBundle\Sensor\SensorFetcherInterface;  
  
class DummySensorFetcher implements SensorFetcherInterface  
{  
 public function supports(SensorModel $sensor): bool  
 {  
 return $sensor->sensorSource === 'dummy';  
 }  
 Verwende ich i.a nie  
 public function fetch(SensorModel $sensor): ?array  
 {  
 return [  
 'sensorID' => $sensor->sensorID,  
 'sensorValue' => rand(0, 100),  
 'sensorEinheit' => 'dummy',  
 'sensorValueType' => 'int',  
 'sensorSource' => $sensor->sensorSource,  
 ];  
 }

public function fetchArr(array $sensors): ?array // neue Methode

{

$res=array();

try {

if (count($sensors) > 0) {

$url=$sensors[0]->geraeteUrl;

if ($url && !str\_starts\_with($url, 'http://') && !str\_starts\_with($url, 'https://')) {

$url = 'http://' . $url;

}

}

if (empty($url)) {

$message = "Heizstab: keine url Sensor {$sensors[0]->sensorID}";

$this->connection->update('tl\_coh\_sensors', [

'lastError' => $message

], ['id' => $sensor->id]);

$this->logger->debugMe($message);

return null;

}

if ( $this->getDataFromDevice($url) === null) {

return null;

}

// Zugriff auf Werte, z.B.:

foreach ($sensors as $sensor) {

$lokalAccess=$sensor->sensorID;

if (!empty($sensor->sensorLokalId)) $lokalAccess=$sensor->sensorLokalId;

$value = $this->getHeizstabdata($lokalAccess);

$einheit=$sensor->sensorEinheit;

if (!empty($sensor->transFormProcedur)) {

if (method\_exists($this, $sensor->transFormProcedur)) {

$arr = $this->{$sensor->transFormProcedur}($value);

$einheit=$arr['einheit'];

$value=$arr['wert'];

} else {

}

}

if ($value === null) {

} else {

$res[$sensor->sensorID] = [

'sensorID' => $sensor->sensorID,

'sensorValue' => $value,

'sensorEinheit' => $sensor->sensorEinheit,

'sensorValueType' => $sensor->sensorValueType,

'sensorSource' => $sensor->sensorSource,

];

$this->connection->update('tl\_coh\_sensors', [

'lastUpdated' => time(),

'lastValue' => $value,

'lastError' => '',

], ['id' => $sensor->id]);

}

}

return $res;

} catch (\Throwable $e) {

$message = "Heizstab: Fehler bei : " . $e->getMessage();

$this->connection->update('tl\_coh\_sensors', [

'lastError' => $e->getMessage()

], ['id' => $sensor->id]);

return null;

}

return $res;

}  
}

# 4. Resources/config/services.yaml

services:  
 \_defaults:  
 autowire: true  
 autoconfigure: true  
  
 Acme\AcmeSensorExtensionBundle\Sensor\DummySensorFetcher:  
 tags: ['sensor.fetcher']

# 5. composer.json

{  
 "name": "acme/sensor-extension-bundle",  
 "description": "Zusatz-Fetcher für ContaoHab Sensor-System",  
 "type": "contao-bundle",  
 "autoload": {  
 "psr-4": {  
 "Acme\\AcmeSensorExtensionBundle\\": "src/Acme/AcmeSensorExtensionBundle/"  
 }  
 },  
 "extra": {  
 "contao": {  
 "bundles": {  
 "Acme\\AcmeSensorExtensionBundle\\AcmeSensorExtensionBundle": ["all"]  
 }  
 }  
 },  
 "require": {  
 "php": "^8.1",  
 "pbdkn/contao-contaohab-bundle": "^1.0"  
 }  
}

# Zusammenfassung

Das Beispiel zeigt, wie ein eigenständiges Contao-Bundle entwickelt werden kann, das zusätzliche SensorFetcher registriert.  
Es kann flexibel in jedes Contao-Projekt integriert werden, ohne Änderungen am Hauptbundle COH vorzunehmen.