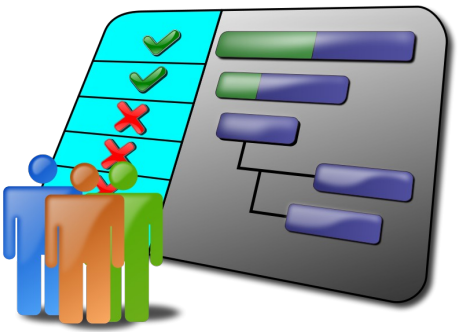
Plugin-Entwicklung

Für Fortgeschrittene?



Heutige

Agenda

- Übliche Probleme
  - Entschlackung
  - Unit-Tests
  - Komponenten
- Mögliche Lösungen
- CLI-Tools

Bestandsaufnahme

Überlaufende
Bootstrap

- viele Event-Subscriber
- viel Logik

- schlecht wartbar
- unübersichtlich
- kaum testbar

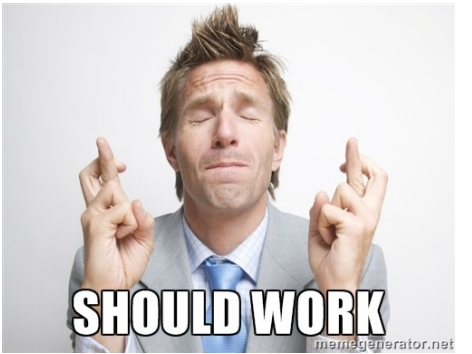
Wohin mit den

Komponenten?

- Zend_Namespaced_Classes
- \Php\Namespaces
- ~~require / include~~
- Shopware()->Resource()
- Abhängigkeiten?

Was ist mit

Tests?



Gegenmaßnahmen

Einsetzen von

Subscribern

- Seit 4.1.4
- Auslagerung von Event-Listenern
- Kompatibel mit dem Symfony SubscriberInterface

Subscriber: Registrierung

engine\Shopware\Plugins\Local\Frontend\SwagMyPlugin\Bootstrap.php

```php
private function createEvents()
 {
     $this->subscribeEvent(
         'Enlight_Controller_Front_DispatchLoopStartup',
         'onStartDispatch'
     );
 }
```

Subscriber: Registrierung

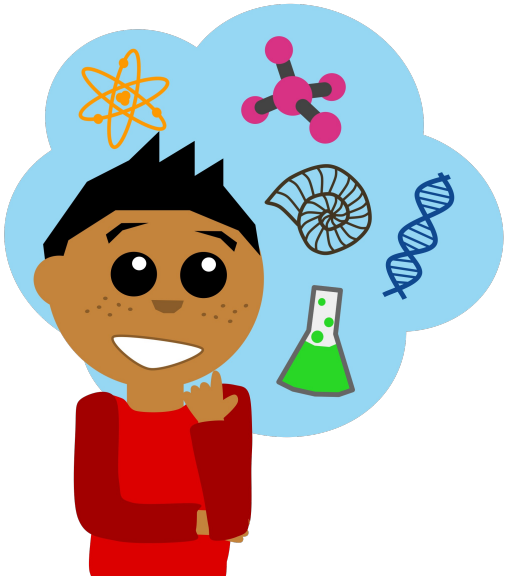
engine\Shopware\Plugins\Local\Frontend\SwagMyPlugin\Bootstrap.php

```php
public function onStartDispatch(Enlight_Event_EventArgs $args)
{

    $subscribers = array(
        new \Shopware\SwagMyPlugin\Subscriber\Checkout()
    );

    foreach ($subscribers as $subscriber) {
        $this->Application()->Events()->addSubscriber($subscriber);
    }
```

Subscriber: Implementierung

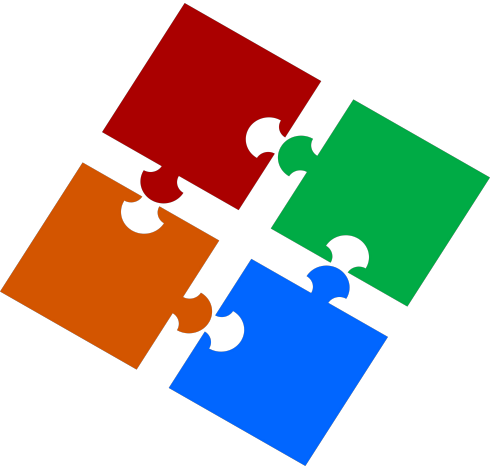
engine\Shopware\Plugins\Local\Frontend\SwagMyPlugin\Subscriber\Checkout.php

```php
class Checkout implements \Enlight\Event\SubscriberInterface
{
    public static function getSubscribedEvents()
    {
        return array(
            'Enlight_Controller_Action_PostDispatchSecure_Backend_Order' => 'extendOrderModule',
        );
    }

    public function extendOrderModule(\Enlight_Event_EventArgs $arguments)
    {
        /** @var \Enlight_Controller_Action $subject */
        $subject = $arguments->getSubject();
        $request = $subject->Request();
        $view = $subject->View();

        // your code
    }
}
```



Was sind die
Vorteile?

- Entschlackung der Bootstrap
- Lesbarkeit / Wartbarkeit
- Registrierung von Events ohne Neuinstallation
- Separation of concerns



Umgang mit

Komponenten

- Wo ablegen?
- Wie registrieren?
- Wie global verfügbar machen?

Komponenten: Namespace registrieren

engine\Shopware\Plugins\Local\Frontend\SwagMyPlugin\Bootstrap.php

```php
public function afterInit()
{
    $this->Application()->Loader()->registerNamespace(
        'Shopware\Plugins\SwagScdExample',
        $this->Path()
    );
}
```

Komponenten: Namespace registrieren

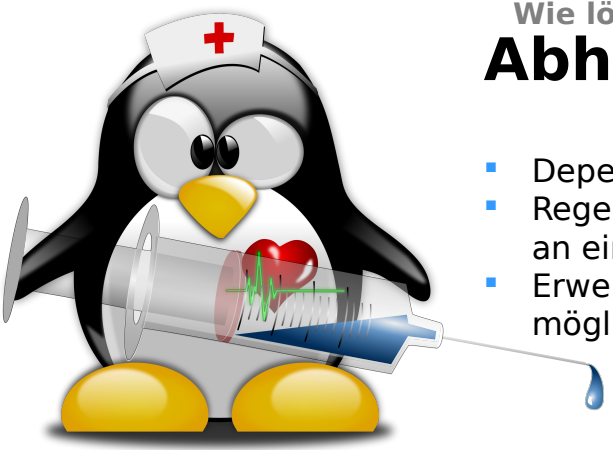
engine\Shopware\Plugins\Local\Frontend\SwagMyPlugin\Controllers\Frontend\Test.php

```php
namespace Shopware\Plugins\SwagScdExample\Components;

use Shopware\Components\Model\ModelManager;

class MyComponent
{
    private $em;

    public function __construct(ModelManager $em)
    {
        $this->em = $em;
    }
```

Komponenten: Namespace registrieren

engine\Shopware\Plugins\Local\Frontend\SwagMyPlugin\Components\MyComponent.php

```php
public function indexAction()
{
    $myComponent = new \Shopware\Plugins\SwagScdExample\Components\MyComponent(
        $this->get('models')
    );
    $myComponent->doStuff();
}
```



Wie löse ich
Abhängigkeiten?

- Dependency Injection
- Regelung der Abhängigkeiten an einem Ort
- Erweiterung über Event-System möglich

Komponenten: Abhängigkeiten

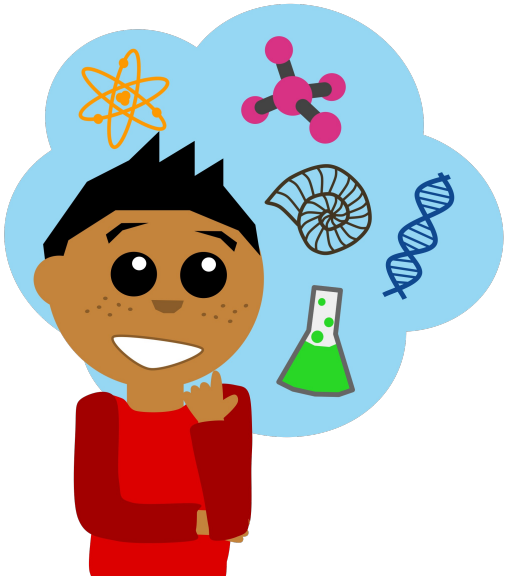
engine\Shopware\Plugins\Local\Frontend\SwagScdExample\Subscriber\Resources.php

```php
class Resources implements \Enlight\Event\SubscriberInterface
{
    public static function getSubscribedEvents()
    {
        return array(
            'Enlight_Bootstrap_InitResource_SwagScdExample_MyComponent'
                => 'onInitResourceMyComponent',
            'Enlight_Bootstrap_InitResource_SwagScdExample_AnotherComponent'
                => 'onInitResourceAnotherComponent'
        );
    }
}
```

Komponenten: Abhängigkeiten

engine\Shopware\Plugins\Local\Frontend\SwagScdExample\Subscriber\Resources.php

```php
class Resources implements \Enlight\Event\SubscriberInterface

    public function __construct(Container $container) { ... }
    public static function getSubscribedEvents() { ... }

    public function onInitResourceMyComponent()
    {
        return new MyComponent($this->container->get('models'));
    }

    public function onInitResourceAnotherComponent()
    {
        return new AnotherComponent(
            $this->container->get('models'),
            $this->container->get('swagscdexample_mycomponent')
        );
    }
```



Kleines
Fazit

- Namespaces nutzen
- require / include vermeiden
- Auf Shopware()-Zugriffe verzichten
- DI-Container kann über Event-System angebunden werden
- Zukünftig: Eigene services.xml

Plugins

Testen

- Qualitätssicherung
- Finden von Regressionen
- Probleme:
  - Abhängigkeiten
  - Globaler State
  - Datenbank

Unittest: Ausgangslage

engine\Shopware\Plugins\Local\Frontend\SwagScdExample\Bootstrap.php

```php
public function beforeCheckout(\Enlight_Event_EventArgs $args)
{
    /** @var \Enlight_Controller_Action $controller */
    $controller = $args->get('subject');
    $request = $controller->Request();

    if ($request->getActionName() != 'finish') {
        return;
    }
    $amount = Shopware()->Modules()->Basket()->sGetAmount();
    $customerGroup = Shopware()->Session()->sUserGroup;


    if ($customerGroup == 'EK' && $amount * 1.1 <= 50) {
        $controller->forward('confirm');
    }
}
```

Unittest: Alternative

engine\Shopware\Plugins\Local\Frontend\SwagScdExample\Subscriber\Basket.php

```php
public static function getSubscribedEvents() { ... }

public function beforeCheckout(\Enlight_Event_EventArgs $args)
{
    /** @var \Enlight_Controller_Action $controller */
    $controller = $args->get('subject');
    $request = $controller->Request();

    if ($request->getActionName() != 'finish') {
        return;
    }

    if (!$this->checkoutAllowed()) {
        $controller->forward('confirm');
    }
}
```

Unittest: Alternative

engine\Shopware\Plugins\Local\Frontend\SwagScdExample\Subscriber\Basket.php

```php
public static function getSubscribedEvents() { ... }

public function beforeCheckout() { ... }

public function checkoutAllowed()
{
    if ($this->customerGroup == 'EK' && $this->amount * 1.1 <= 50) {
        return false;
    }

    return true;
}
```

Unittest: phpunit.xml

engine\Shopware\Plugins\Local\Frontend\SwagScdExample\phpunit.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<phpunit bootstrap="../../../../../../tests/Shopware/TestHelper.php">
<testsuite name="SwagScdExample Test Suite">
    <directory>tests</directory>
</testsuite>
</phpunit>
```

Unittest: Plugins laden

engine\Shopware\Plugins\Local\Frontend\SwagScdExample\tests\Test.php

```php
class PluginTest extends Shopware\Components\Test\Plugin\TestCase
{
    /*
     * Ensure that the plugin is installed and loaded
     */
    protected static $ensureLoadedPlugins = array(
        'SwagScdExample' => array()
    );
}
```

Unittest: Namespace registrieren

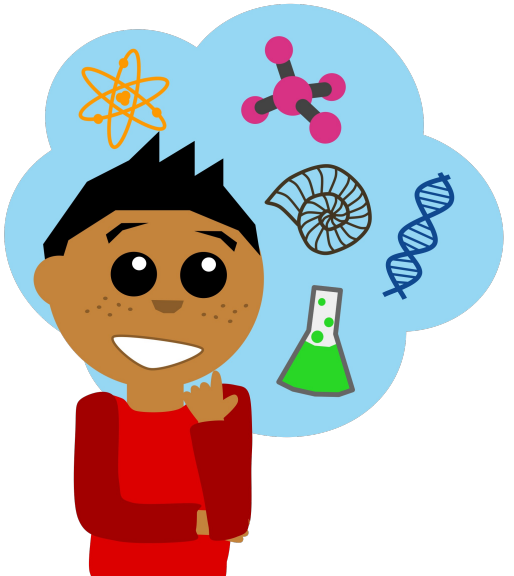
engine\Shopware\Plugins\Local\Frontend\SwagScdExample\tests\Test.php

```php
class PluginTest extends Shopware\Components\Test\Plugin\TestCase
{
    protected static $ensureLoadedPlugins = array(...);

    public static function setUpBeforeClass()
    {
        parent::setUpBeforeClass();

        $helper = \TestHelper::Instance();
        $loader = $helper->Loader();

        $loader->registerNamespace(
            'Shopware\Plugins\SwagScdExample',
            dirname(__DIR__) . '/'
        );
    }
}
```

Unittest: Test implementieren

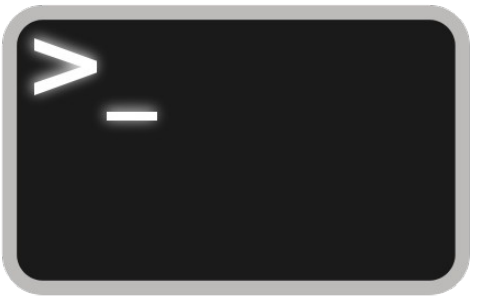
engine\Shopware\Plugins\Local\Frontend\SwagScdExample\tests\Test.php

```php
class PluginTest extends Shopware\Components\Test\Plugin\TestCase
{
    protected static $ensureLoadedPlugins = array(...);
    public static function setUpBeforeClass() { ... }

    public function testCheckoutAllowedShouldAllow()
    {
        $subscriber = new \Shopware\Plugins\SwagScdExample\Subscriber\Basket(50, 'EK');
        $result = $subscriber->checkoutAllowed();

        $this->assertEquals(true, $result);
    }
}
```



Kleines
Fazit

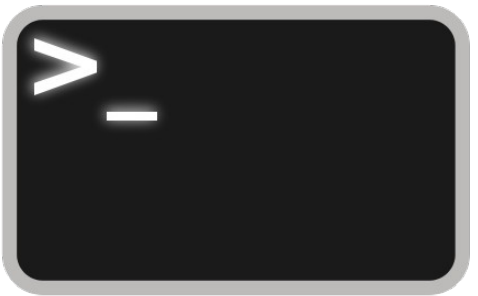
- Logik möglichst atomar
- Globalen State immer injecten
- Über Shopware-Testsuite stehen viele Helfer zur Verfügung



Die Shopware

CLI-Tools

- Setzen auf den Symfony Console Tools auf
- „Shopware für die Kommandozeile“
- Einfach zu integrieren



Was ist damit

möglich?

- Plugins (de)installieren
- Generierung von Thumbnails
- Leeren des Caches
- Snippet-Handling

CLI-Tools: Benutzen

Anfrage:
php bin/console sw:plugin:install SwagLiveShopping

Antwort:
The plugin SwagLiveShopping is already installed.

Anfrage:
php bin/console sw:plugin:config:set SwagLiveShopping displayRating 0

Antwort:
Plugin configuration for Plugin SwagLiveShopping saved.

CLI-Tools: Registrieren

engine\Shopware\Plugins\Local\Frontend\SwagScdExample\Bootstrap.php

```php
public function install()
{
    $this->subscribeEvent(
        'Shopware_Console_Add_Command',
        'onAddConsoleCommand'
    );
    [...]
}

public function onAddConsoleCommand()
{
    $this->registerDefaultSubscriber();

    return new SwagScdExampleCommand();
}
```

CLI-Tools: Implementieren

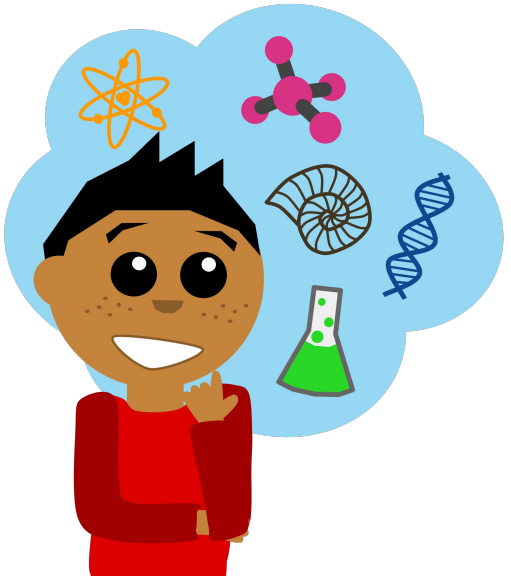
engine\Shopware\Plugins\Local\Frontend\SwagScdExample\Commands\SwagScdExampleCommand.php

```php
class SwagScdExampleCommand extends ShopwareCommand
{
    protected function configure()
    {
        $this->setName('swagscdexample:example')
            ->setDescription('Run example command.')
            ->addArgument(
                'name',
                InputArgument::REQUIRED,
                'Your name'
            )
        ;
    }
}
```

CLI-Tools: Implementieren

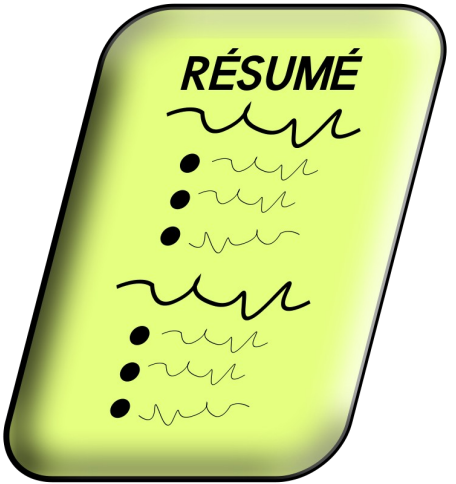
engine\Shopware\Plugins\Local\Frontend\SwagScdExample\Commands\SwagScdExampleCommand.php

```php
class SwagScdExampleCommand extends ShopwareCommand
{
    protected function configure() { ... }

    protected function execute(InputInterface $input, OutputInterface $output)
    {
        $name = $input->getArgument('name');

        $output->writeln("<info>{$name}</info>");

        $this->container->get('swagscdexample_mycomponent')->doStuff();
    }
}
```



Kleines
Fazit

- SW über Konsole ansprechen
- Wiederverwendung bestehender Komponenten



Großes
Fazit

- Use it!
- Plugins als Mini-Applikation in Shopware
- Beispiel bei GitHub