Shopware

Die dritte Shopsoftware im Bunde hat erst vor einem Jahr den Sprung in die Open-Source-Welt vollzogen. Seit Oktober 2010 gibt es die Community Edition von Shopware, die auf dem so genannten Enlight-Framework basiert, das wiederum auf dem Zend Framework aufgebaut ist. Von Letzterem wurden das MVC-Design-Pattern sowie die Request- und Response-Objekte geerbt. Das auf Events und Hooks basierende Plug-in-System ist eine Eigenentwicklung. Wie bei den oben genannten Systemen werkelt auch bei Shopware eine MySQL-Datenbank im Hintergrund. Die Templates basieren auf *Smarty* 3. Um dem Backend ein Windows Look and Feel zu geben, kommt dort *ExtJS* 4 zum Einsatz.

Applikationsstruktur

Auch Shopware vereinfacht die Entwicklungsarbeit durch eine sinnvolle Struktur der Applikation und enthält Model-, View- und Controller-Elemente.

Das Verzeichnis */engine/Shopware/* enthält die wichtigsten Teile der Anwendung, nämlich zum einen die Controller in */Controllers*, die weiter in */Backend* und */Frontend* aufgeteilt sind, und zum anderen die Model-Klassen in */Models*. Last but not least liegen die Output-relevanten Dateien der Smarty Engine in */templates*. Funktionserweiterungen durch neue Module bezeichnet man im Shopware-Kontext als *Plug-ins*. Diese werden in einem separaten Verzeichnis gespeichert (*/engine/Shopware/Plugins*) und liegen in zwei unterschiedlichen Codepools vor: */Default* und */Local*. Während im Ersteren alle Core-Module gespeichert sind, ist Letzterer genau der richtige Ort für eigene Erweiterungen.

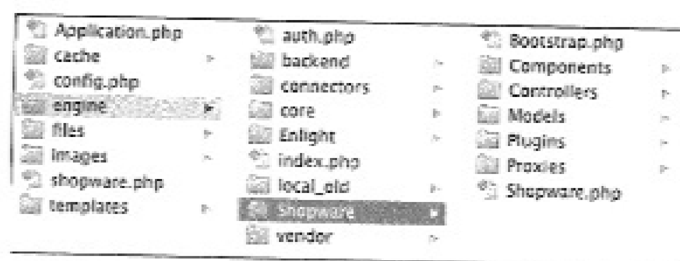

Abb. 5: Die Shopware Struktur

Ein eigenes Plug-in erstellen

Wir erstellen in diesem Fall gemeinsam ein kleines Plug-in, das uns erlaubt, einen Text im Backend einzutragen, der an einer bestimmten Stelle im Frontend dargestellt wird. Um ein neues Plug-in zu erstellen, legen Sie zunächst einen Ordner innerhalb von */engine/Shopw[...] Plugins/Local/* an. Darin erstellen Sie eine Datei *Bo[...]strap.php*. sie ist für jedes neue Plug-in obligatoris[...] In dieser Datei ist unter anderem die Installationsro[...]tine enthalten – handelt es sich bei Shopware doch [...] das einzige System der drei vorgestellten, bei der [...] externes Modul einfach über einen 1-Click-Installer [...]stalliert werden kann. Die erste Methode *install()* w[...] also immer dann aufgerufen, wenn ein Plug-in über d[...] Backend installiert wird.

Über *createEvent()* wird ein neues Event erstellt. A[...]ßerdem erzeugt *onPostDispatch* einen neuen lokal[...] Listener. Als Nächstes wird das gerade erstellte Eve[...] über *subscribeEvent()* in der Datenbank persistier[...] Schließlich muss sichergestellt werden, dass der Retur[...] Wert *true* ausgegeben wird, damit die Plug-in-Installa[...]tion als erfolgreich ausgegeben wird. Das *Form*-Objek[...] wird in diesem Beispiel verwendet, um eine Text-Are[...] zu erzeugen, die im Backend mit beliebigem Text gefül[...] werden kann. Die zweite Methode namens *onPostDis[...]patch()* in unserer Bootstrap-Datei enthält den Listene[...] auf das Post-Dispatch-Event, das bei jedem Seitenaufru[...] des Frontends angestoßen wird (Listing 7).

Das Objekt *Enlight_Event_EventArgs $args* erlaub[...] es uns in diesem Fall, auf die Event-Eigenschaften zu[...] zugreifen. Mithilfe von *getSubject()* ist es möglich, a[...] die Referenz des Objekts zuzugreifen, in das das Even[...] gelegt wurde, in unserem Fall also den Front Controller[...] Auf diese Weise lässt sich also sowohl auf die Controlle[...] als auch die Request- und Response-Objekte zugreifen[...] Last but not least sorgen wir dafür, dass die zentrale[...] Config-Ressource *$config* genutzt wird, um die gerade[...] gefüllte Variable *$config->yourtext* auszulesen bzw. [...] Template zur Verfügung zu stellen.

Listing 6

```
<?php
class Shopware_Plugins_Frontend_PHPM_Bootstrap
  extends Shopware_Components_Plugin_Bootstrap
{
  public function install()
  {
   $event = $this->createEvent('Enlight_Controller_Action_PostDispatch','onPostDispatch');
   $this-> subscribeEvent($event);

   $form = $this->Form();
   $form->setElement('textarea', 'yourtext',
     array('label'=>'Text for left column','value'=>'Hello World'));
   $form->save();
   return true;
  }
```

Listing 7

```
public static function onPostDispatch(Enlight_Event_EventArgs $args)
  {
   $view = $args->getSubject()->View();
   $config = Shopware()->Plugins()->Frontend()->PHPM()->Config();
   $view->pluginText = $config->yourtext;
```

Fazit

Die drei kurzen Beispiele haben Ihnen im Schnellverfah[...]ren einen Einblick in die Modulsysteme von Magento[...] OXID eShop und Shopware gegeben. Während sich Magento und Shopware aufgrund Ihrer Zend-Frame[...]work-Herkunft diesbezüglich ähneln und von Code[...]pools sowie Events bzw. Listenern profitieren, geht OXID eShop einen eigenen Weg und führt transparente Klassen ein, um multiple Vererbung in PHP zu reali[...]sieren.

Dr. Roman Zenner arbeitet als freiberuflicher Autor und Bera[...] im E-Commerce-Umfeld und hat in diesem Zusammenhang sch[...] mehrere Fachbücher zu Magento und OXID eShop bei O'Reilly ve[...]öffentlicht. Unter dershopberater.de berät er Kunden bei der Aus[...]wahl des passenden Shopsystems. Außerdem bloggt er a[...] ecommpunk.com über aktuelle Themen der E-Commerce-Szene.