Ausgabe 5.13 August/September www.phpmagazin.de S&S

PHPmagazin

PHPmagazin

Deutschland 9,80 € Österreich 10,80 € | Schweiz 19,20 sFr
Niederlande 11,25 € | Luxemburg 11,25 €

PHP • JavaScript • Open Web Technologies

RabbitMQ
Geheimnisse des asynchronen Nachrichtenversands

Single-Page-Apps
Die Konzepte dahinter

PHP-Security
Die Evolution seit PHP 4.x

Legacy-Anwendungen
Sicherer Umgang mit Altlasten

©istockphoto.com/iatsun

Chrome Developer Tools
Werkzeugkasten für WWW-Arbeiter

Magento Testsuites
Unit Testing wie die Profis

4 195649 609809 05

Workflow-gestützte Datenbankanwendungen mit dem Limbas-Framework

Workflows in Limbas

Limbas ist ein PHP-Open-Source-Framework, das die Umsetzung von Datenbankanwendungen ermöglicht. Es stellt verschiedene Module bereit, die nach individuellen Anforderungen konfiguriert und erweitert werden können. Eine große Neuerung der Ende März veröffentlichten Version 2.5 war die integrierte Workflow Engine. In diesem Artikel soll gezeigt werden, wie ein Workflow in Limbas realisiert werden kann.

PHP-Magazin-5.2013

von Armin Litzl

Des Weiteren werden einige grundlegende Module verwendet, die im Zusammenhang von Bedeutung sind. Hierzu gehört auch der Formulareditor, mit dem Formulare für die Bearbeitung von Datensätzen gestaltet werden können. Diese Formulare enthalten gleichzeitig die Benutzerschnittstelle für Workflow-Funktionalitäten. Außerdem wird gezeigt, wie über Erweiterungsschnittstellen auf die eingebaute Kalenderfunktionalität zugegriffen wird und Wiedervorlagen zur zeitlichen Erinnerung von zu erledigenden Aufgaben eingesetzt werden können.

Ein Workflow zur Kundenakquise

Im folgenden Beispiel wird ein Workflow zur Akquise von Neukunden für Softwarelösungen vorgestellt. Anschließend wird gezeigt, wie dieser in Limbas umgesetzt werden kann. Der Workflow ist in **Abbildung 1** schematisch dargestellt.

> **Hinweis**
>
> Der hier beschriebene Workflow ist Teil der Demo zur neuen Limbas-Version 2.6. Diese kann von www.limbas.com heruntergeladen werden.

Um einen möglichen Neukunden anzuwerben, sendet man ihm zunächst ein Werbeprospekt zu (1). Wenn der Kunde daraufhin Interesse an dem Produkt zeigt, dann wird ihm außerdem ein Onlinedemozugang bereitgestellt (2). Bekräftigt der Kunde weiterhin sein Interesse, dann besteht noch die Möglichkeit, ihm eine Demo-CD

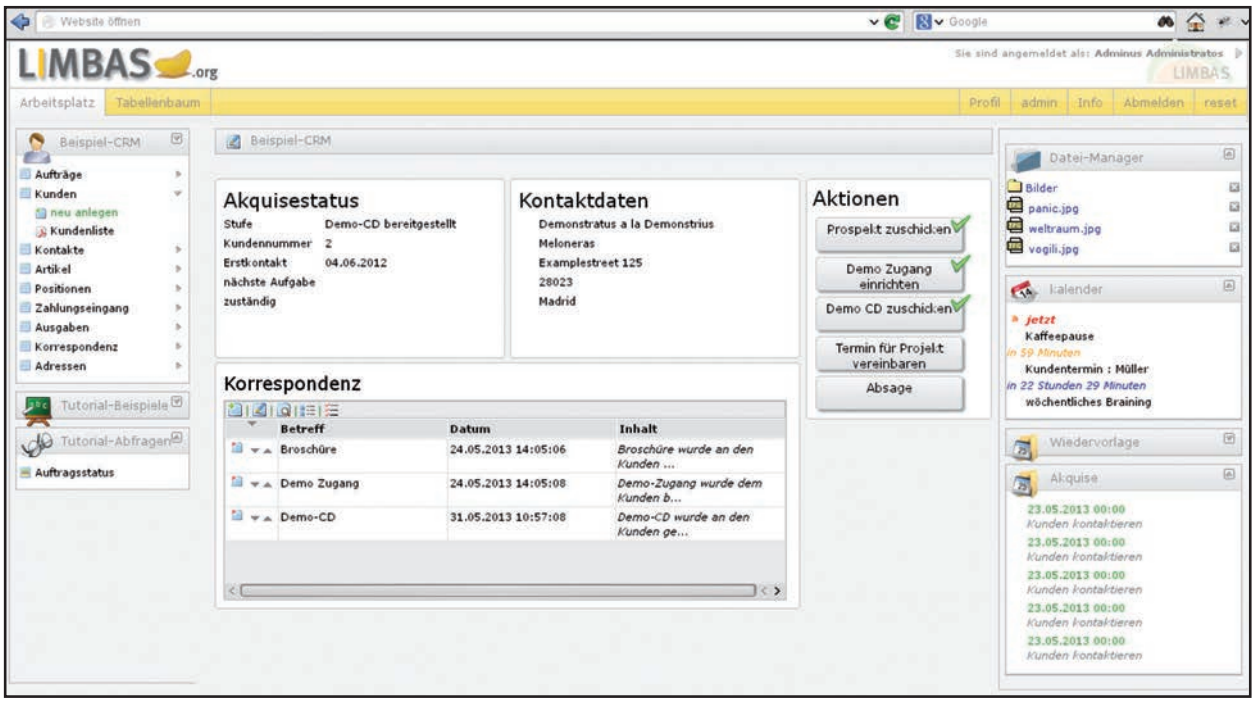

Abb. 1: Ein Workflow für die Akquise von Neukunden, die orangefarbigen Flusselemente heben die von dem Workflow erstellten Datenobjekte in Limbas hervor

zukommen zu lassen (3). Wenn der Kunde endgültig überzeugt ist, dann sollte ein Projekttermin mit ihm vereinbart werden (4). Falls der Kunde jedoch eine Absage erteilt, wird der Kundenakquise-Workflow abgebrochen (5).

Der Workflow definiert die mögliche Abfolge der einzelnen Schritte in der Kundenakquise. Wurde der Workflow initiiert, dann ist der erste Schritt immer die Zusendung eines Werbeprospekts. Außerdem soll es in diesem Beispiel erst möglich sein, einen Projekttermin zu vereinbaren, wenn der Kunde einen Demozugang oder eine Demo-CD erhalten hat. Ein Abbruch der Kundenakquise ist bis vor der Fixierung des Projekttermins möglich.

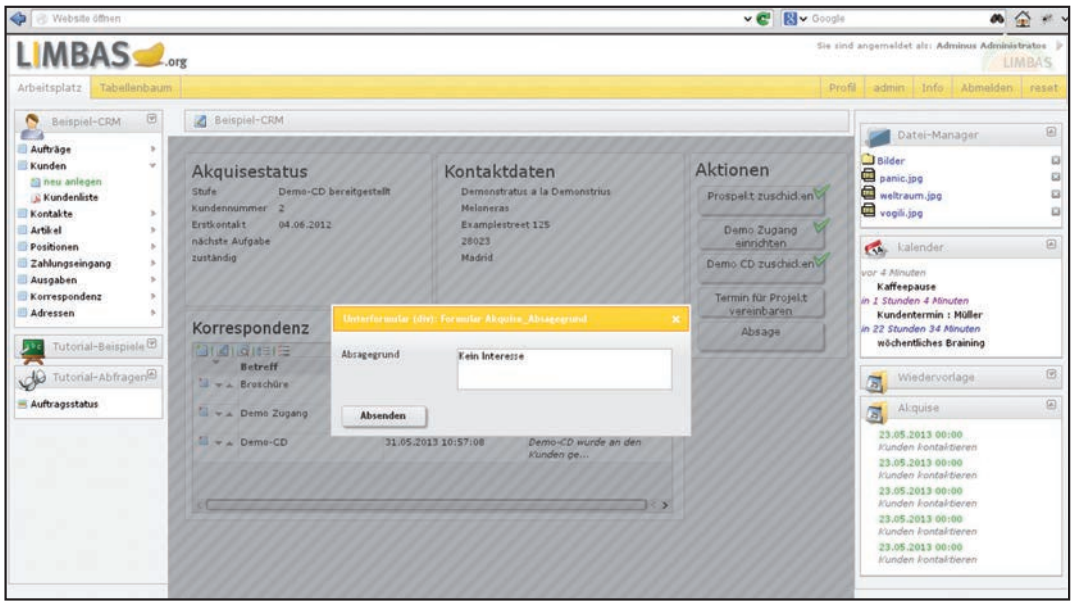

Abb. 2: Unterformular „Akquise_Absagegrund“

Des Weiteren legt der Workflow nach Durchführung jedes Schritts einen Eintrag für die Korrespondenz mit dem Kunden an. Um den Mitarbeiter nach jeder Akquiseaktion daran zu erinnern, den Kunden wieder nach sieben Tagen zu kontaktieren und sich nach seinem Interesse zu erkundigen, wird hierzu automatisch eine Wiedervorlage in Limbas angelegt.

Zuletzt nimmt der Workflow Benutzereingaben über Formulare entgegen und trägt sie in die entsprechenden Datensätze ein. So wird z. B. der Projekttermin in den Kalender eingetragen oder der für die Absage vom Benutzer angegebene Absagegrund in einen automatisch erstellten Korrespondenzeintrag übernommen. **Abbildung 2** zeigt das Unterformular *Akquise_Absagegrund*.

Anlegen des Workflows

Der Workflow muss zuerst im Admininterface definiert werden. Hierzu erstellen wir einen neuen Workflow mit dem Namen *Kundenakquise*. Nach Klick auf den Editier-Button können die einzelnen Schritte (Tasks) des Workflows definiert werden. Zu jedem Task wird außerdem die Tabelle angegeben, die das Formular für die entsprechende Workflow-Aktion enthalten soll. In unserem Beispiel ist dies für alle Tasks die Tabelle *Kunden*, da der Nutzer einen Datensatz der Kundentabelle aufrufen und hier über ein Formular die einzelnen Aktionen des Workflows durchführen soll.

Das Admininterface bietet auch die Möglichkeit, Parameter an den Workflow zu übergeben. Auf diese Weise können konfigurierbare Workflows entwickelt werden. So übergeben wir z. B. für Task 1 bis 3 als Parameter ein Array mit dem Element *reminder_days*. Dieses legt fest, wie viele Tage nach der Zusendung einer Broschüre, der Bereitstellung einer Demo-CD oder eines Demozugangs der Benutzer über eine Wiedervorlage daran erinnert werden soll, den Kunden wieder zu kontaktieren.

Um Wiedervorlagen, die automatisch von dem Workflow nach Durchführung einer Aktion angelegt werden sollen, separat anzuzeigen, benötigen wir eine neue Wiedervorlagenliste. Diese kann über den Menüpunkt WIEDERVORLAGE im Admininterface angelegt werden. Hierfür wählen wir den Namen *Akquise* und legen als Grundtabelle ebenfalls *Kunden* fest.

Implementierung des Workflows

> **Hinweis**
>
> Der vollständige Quellcode kann von der SourceForge-Projektseite heruntergeladen und getestet werden (www.limbas.org).

Ein Workflow wird in Limbas als Erweiterung gehandhabt. Wir haben für das im Anschluss vorgestellte Skript eine Datei mit dem Namen *ext_gtab_change.inc* im Ordner *dependent/EXTENSIONS/workflow/* angelegt. Diese Datei beinhaltet unsere individuellen Workflow-Funktionen und die Initiierung des Workflows, die im Folgenden dargestellt ist:

```
# limbas workflow engine
require_once("extra/workflow/lwf.lib");
# get wfl instance of dataset
$wfl_inst = lmb_wfl_getRecordInst(1,$gtabid,$ID);
# start Workflow
if(isset($myExt_sendTask)){
  # init Task
  lmb_wfl_init(1,$wfl_inst,$wfl_task,$gtabid,$ID,$params);
}
```

Zunächst müssen wir die Bibliothek *lwf.lib* einbinden, die die Workflow Engine von Limbas bereitstellt. Zusätzlich benötigen wir drei grundlegende Informationen:

1. *$wfl_id ID* des gewünschten Workflows: Generell können beliebig viele Workflows erstellt werden. Es wird aber nur immer ein einziger Workflow auf einmal abgearbeitet. Unser Beispiel enthält nur einen Workflow mit dem Namen Akquise und der ID 1.
2. *$wfl_task ID* des auszuführenden Tasks: Die Task-ID entscheidet, welcher Task als Nächstes ausgeführt werden soll. Diese kann zum Beispiel über ein Formular übergeben werden. In unserem Fall wird über ein Formular der Parameter *$myExt_sendTask* übergeben.
3. *$wfl_inst ID* der aktuellen Instanz: Limbas kann jedem Datensatz einer Tabelle eine oder mehrere Instanzen (Workflowketten) zuweisen. Ob ein Datensatz einer Instanz zugewiesen ist, kann über die Hilfsfunktion *lmb_wfl_getRecordInst* abgefragt werden. Ebenso wäre es möglich, die Instanz-ID über Formulare oder URLs mit zu übergeben. *LIMBAS* übergibt in internen Requests standardmäßig die Workflow-ID (*$wfl_id*) und Instanz-ID (*$inst_id*) falls vorhanden. In unserem Fall fragen wir explizit nach der Instanz-ID für einen bestimmten Kunden.

Die Funktion *lmb_wfl_init* initiiert schließlich den Workflow. Dabei werden der Funktion zusätzliche Variablen als Parameter übergeben:

- *$gtabid* ist die ID der aktuellen Tabelle
- *$ID - ID* ist die ID des entsprechenden Datensatzes
- *$params* – benutzerspezifischer Parameter

Task-Funktionen

Nach der Initiierung versucht Limbas, den entsprechenden Task auszuführen. Dazu benötigen wir für jeden Task eine entsprechende Funktion. Diese Funktion muss den Namen *lmbWfl_[TASK_ID]* tragen, wobei *TASK_ID* mit der entsprechenden Task-ID zu ersetzen ist. Diese Funktionen sollen in zukünftigen Limbas-Versionen teilweise durch ein grafisches Frontend erstellbar sein. Ob der Workflow immer nur den nächsthöheren Task oder beliebige Tasks ausführen darf, liegt ganz in der Hand des Entwicklers. In unserem Beispiel kann jeder Task aufgerufen werden, damit der Akquisestatus frei änderbar bleibt. Fangen wir mit dem Task 0 bzw. des Starten des Workflows an (Listing 1).

Listing 1

```
function lmbWfl_0($wfl_inst, $active_task,
  $gtabid = null, $ID = null, $params = null) {
  if ($params['showElements']) {
    $hist = lmb_wfl_getHistory(1, $wfl_inst);
    myExt_showElements($hist);
  } else {
    // start workflow
    return 0;
  }
}
```

Unser Task prüft zunächst, ob der Parameter *showElements* übergeben wurde. Dies soll immer der Fall sein, wenn ein Formular dargestellt wird. Dann wird die Hilfsfunktion *myExt_showElements* aufgerufen, die prüft, welche Aktionen der Benutzer durchführen kann und anschließend über eine JavaScript-Funktion das Formular anpasst. In unserem Beispiel übernimmt diese das Einblenden von verfügbaren Aktionsbuttons und das Markieren von bereits erledigten Aktionen.

Falls der Task 0 von dem Benutzer aufgerufen wird, um den Workflow zu initialisieren, dann wird *showElements* nicht übergeben. Erst wenn der Task erfolgreich beendet wurde, ruft der Workflow die nächste höhere Task-ID mit dem Parameter *showElements* auf.

Allgemein gilt, dass ein Task nach erfolgreicher Ausführung als Funktionswert die Task-ID zurückgibt, die als aktiver Task gesetzt wird. In diesem Beispiel setzen wir als aktive Task-ID immer die ID des zuletzt ausgeführten Tasks. Um den Workflow erfolgreich zu initialisieren, wird hier also 0 zurückgegeben. Falls dagegen Fehler während der Ausführung auftreten, soll die Funktion *false* zurückgeben werden. Die Limbas Workflow Engine macht in diesem Fall alle von dem Task durchgeführten Änderungen an der Datenbank wieder rückgängig.

Ein Task kann auch nur *true* zurückgeben, falls er keine Änderungen am Workflow durchgeführt hat, z. B. wenn er mit *showElements* aufgerufen wurde. Der nächste Task ist in unserem Beispiel das Zusenden einer Broschüre an den Kunden (Listing 2).

Limbas protokolliert für alle Instanzen eines Workflows die bereits durchgeführten Tasks in einer History. Damit eine Workflow-Aktion beispielsweise nur einmal durchgeführt wird, prüft man ihr Vorhandensein in der History, die über die Funktion *lmb_wfl_getHistory* abgerufen werden kann. Als Nächstes legen wir eine Wiedervorlage mit der Funktion *myExt_createReminder* an, um den Kunden in dem festgelegten Zeitraum wieder zu kontaktieren und sich nach seinem Interesse zu erkundigen. Diese Hilfsfunktion greift im Wesentlichen auf die von Limbas bereitgestellte Erweiterungsfunktion *lmb_addReminder* zurück. Anschließend wird mit *wflAddCorrespondence* ein Eintrag in der Korrespondenztabelle erstellt. Zuletzt muss noch der Akquisestatus in dem entsprechenden Datenfeld der Kundentabelle neu gesetzt werden. Bei erfolgreicher Ausführung wird die Task-ID 1, ansonsten ein *false* zurückgegeben.

Weitergehende Informationen zu diesen und weiteren Limbas-Basisfunktionen findet man unter www.limbas.org.

Erstellung des Formulars als Benutzerschnitterstelle für den Workflow

Zu guter Letzt erstellen wir noch das entsprechende Akquiseformular für die Durchführung von Workflow-Aktionen. Hierzu rufen wir im Admininterface den Menüpunkt Formulare auf und legen das Formular

Akquise für die Tabelle *Kunden* an. Dieses Formular enthält neben Kundendaten auch einen Bereich für die Workflow-Aktionen (**Abb. 3**).

Um Buttons für die einzelnen Workflow-Aktionen zu erstellen, fügen wir für jede Aktion ein Texteingabefeld mit der entsprechenden Beschreibung ein. Zusätzlich wird dem *onClick*-Attribut der Wert *myExt_sendTask(Id)* zugewiesen, wobei *Id* durch die jeweilige Task-ID ersetzt wird. Diese JavaScript-Hilfsfunktion benutzen wir, um die entsprechenden Task-ID zu übermitteln.

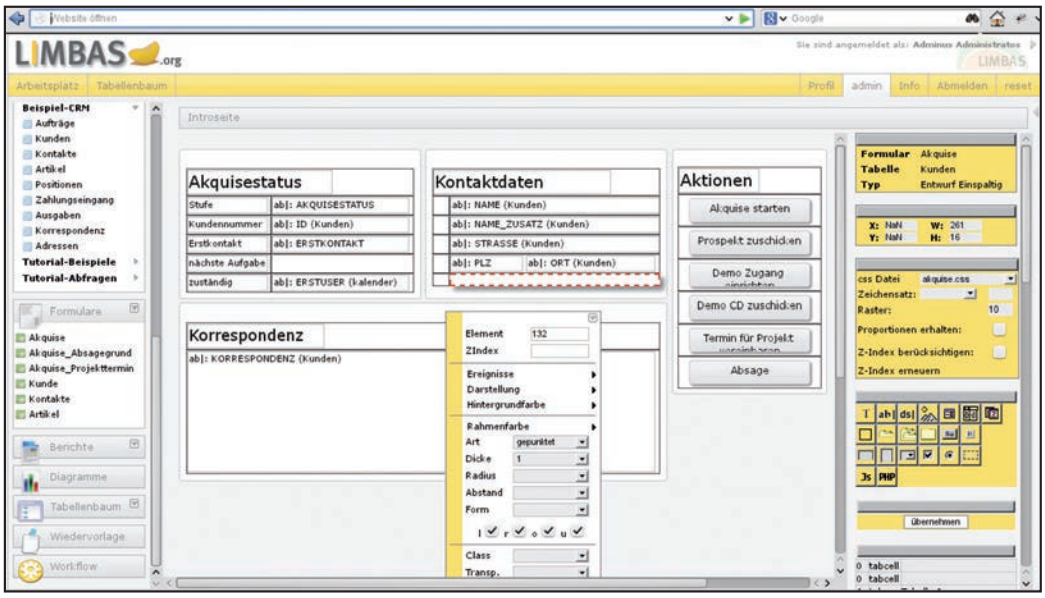

Abb. 3: Erstellung des Formulars

Hinweis

Es ist von Vorteil, die Buttons in eine Tabelle einzufügen. Falls einzelne Workflow-Aktionen versteckt werden sollen, wenn der Benutzer vorhergehende Aktionen noch nicht durchgeführt hat, dann hat dieser Ansatz den optischen Vorteil, dass keine Lücken durch ausgeblendete Buttons erscheinen.

Das spezifische Aus- oder Einblenden der Buttons in den verschiedenen Tasks wird über eine JavaScript-Funktion erledigt, die nach Laden des Formulars ausgeführt wird. Dies hat den Vorteil, auch Workflow-Elemente wie die Aktionsbuttons über den Formulareditor anpassen zu können. Dazu rufen wir die Workflow-Initialisierungsfunktion über ein PHP-Formularelement auf. Diesmal mit gesetztem *showElements*-Parameter.

Hinweis

Das Formularelement sollte in den Vordergrund positioniert werden (hoher Z-Index), damit die zu bearbeitenden Elemente schon existieren.

```
lmb_wfl_init(1,$GLOBALS['wfl_inst'],0,$gtabid,$ID,array('showElements'
                                                        => 1));
```

Fazit

Limbas stellt im Moment alle Mittel bereit, um individuelle Workflows für Datenbankanwendungen in Code zu implementieren. Dieser Ansatz bietet größtmögliche Flexibilität bei der Umsetzung von Workflows, benötigt aber auch fundierte PHP-Kenntnisse, um die entsprechenden Funktionen zu implementieren. In Zukunft ist geplant, dass grundlegende Workflow-Module wie z. B. Wiedervorlagen, E-Mails oder Datenverarbeitungen über einen Workflow-Designer grafisch zusammengestellt werden können.

Armin Litzl: TU Student – arbeitet als Entwickler bei der Limbas GmbH. Kontakt: armin.litzl@limbas.com

Listing 2

```
function lmbWfl_1($wfl_inst, $active_task, $gtabid = null, $ID = null, $params = null) {
  global $gwfl;
  global $session;
  // Get task parameters from admin interface
  $t_params = $gwfl[WFL_ID]['task']['params'];
  // Get history of this workflow instance
  $hist = lmb_wfl_getHistory(WFL_ID, $wfl_inst);
  $days = $t_params['reminder_days'];

  if ($params['showElements']) {
    myExt_showElements($hist);
  } else {
    if (!in_array(1, $hist["task_id"])) {
      // Create reminder to contact customer
      if (!myExt_createReminder(lmb_utf8_decode("Kunden kontaktieren"), $days,
                                                          $gtabid, $ID, $wfl_inst))
        return false;
      // Create entry in correspondance table
      if (!myExt_wflAddCorrespondence($ID, lmb_utf8_decode("Broschüre"),
                          lmb_utf8_decode("Broschüre wurde an den Kunden gesendet.")))
        return false;
      // Set acquisition workflow status
      if (!update_data(array("$gtabid," . ELEMENT_ID_ACQUISITION_STATUS . ",$ID" =>
                                          lmb_utf8_decode("Broschüre zugesandt"))))
        return false;
      return 1;
    }
    return true;
  }
}
```