inkl. CD!

RailsWayCon 2010: Die neusten Infos ab Seite 49

Deutschland € 6,50 Österreich € 7,00 Schweiz sFr 13,40

Infos zur International PHP Conference ab Seite 29

entwickler magazin

entwickler magazin

Software, Systems & Development

www.entwickler-magazin.de | Mai/Juni | 3.10

CD-INHALT:

■ Trial

RemObjects Data Abstract: Das Multi-Tier-gestützte Framework für die Entwicklung plattformübergreifender Datenbankanwendungen. Für .NET, Delphi und Mac OS X.

■ Software

Pentaho Business Intelligence Server CE: Komplette BI-Suite für das Reporting, die OLAP-Datenanalyse oder das Data Mining und die Datenintegration.

Qt SDK: Cross Platform Framework für die Entwicklung Desktop- und Embedded-basierter Anwendungen, inklusive der Qt-Creator-IDE.

Open Limbas: Das webbasierte Framework zur Umsetzung von Unternehmenslösungen und Fachverfahren, inklusive fertiger Basisdienste und Module wie CRM, DMS, CMS, Callcenter, Ticketsystem, Bildarchiv, Workflow und Groupware.

Eclipse GMF: Das Eclipse Graphical Modeling Framework für eine generative Komponenten- und Runtime-Infrastruktur zur Entwicklung grafischer, Eclipse-basierter Editoren für EMF und GEF.

■ Bonus

Zusätzlich finden Sie auf unserer aktuellen Magazin-CD wieder ausgewählte Tools sowie natürlich alle Quellcodes zu den Artikeln im Heft!

▸▸▸ Alle Infos auf Seite 3

Weitere Artikel

4 194156 606509 03

Datenträger enthält Info- und Lehrprogramme gemäß §14 JuSchG

Facebook, iPhone und QT

17 Seiten Spezial

Tools & APIs für Social Developer

Der Web-2.0-Kleber

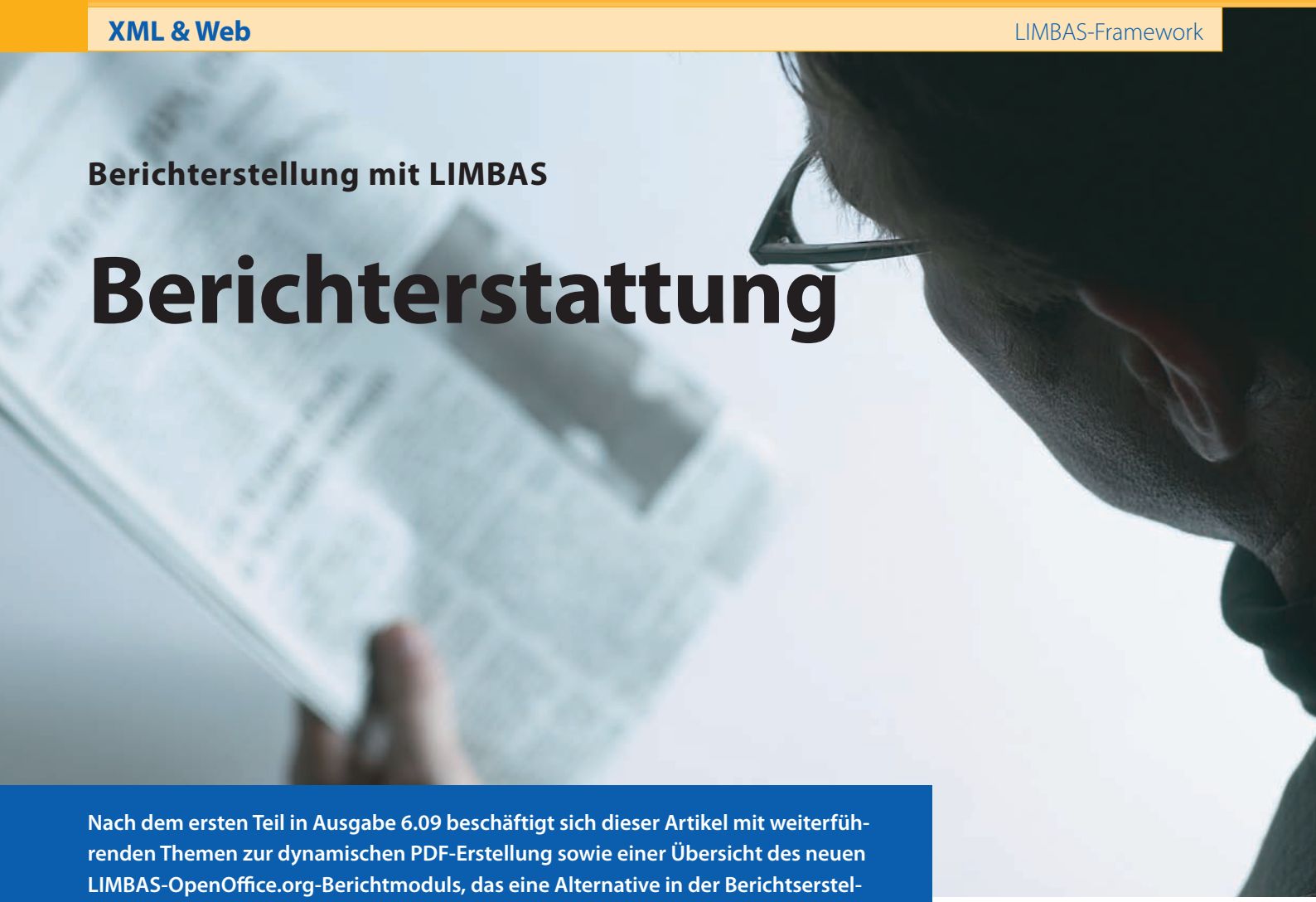
An jQuery gibt es kein vorbei

Herzensangelegenheit

Netzwerkserver in .NET

Scala, jetzt!

Zucker für Java-Developer



Berichterstellung mit LIMBAS

Berichterstattung

Nach dem ersten Teil in Ausgabe 6.09 beschäftigt sich dieser Artikel mit weiterführenden Themen zur dynamischen PDF-Erstellung sowie einer Übersicht des neuen LIMBAS-OpenOffice.org-Berichtmoduls, das eine Alternative in der Berichtserstellung unter LIMBAS darstellt.

von Axel Westhagen

Der erste Teil dieses Artikels [1] hat die grundlegende Erstellung von einfachen PDF-Berichten über das LIMBAS-Framework beschrieben. Dazu gehört unter anderem das Zusammenspiel von Inhalten und Formatierungen sowie deren Erstellung. Mithilfe von FPDF werden die erstellten Berichtsvorlagen in ein PDF umgewandelt, die mit Inhalten aus der jeweiligen LIMBAS-Applikation dynamisch ergänzt werden. Zusammen haben wir als Beispiel ein Produktdatenblatt erzeugt. Es besitzt einen Titel, ein Bild sowie ein Tabellenelement mit den Produktdaten. Der Bericht beschreibt einen einzigen Artikel im Detail. In unserem nächsten Beispiel bauen wir eine Rechnung, die aus den vorhandenen Testtabellen der aktuellen Demoinstallation von LIMBAS gespeist werden soll. Für unseren Rechnungsbericht benötigen wir die Tabellen *Aufträge*, *Kunden*, *Kontakte* und *Positionen*, die in der LIMBAS-Demoinstallation bereits angelegt sind. Die Tabellen *Kunden*, *Kontakte* und *Positionen* sind jeweils mit der Tabelle *Aufträge* verknüpft.

Zusätzlich ist die Tabelle *Kontakte* ebenfalls noch mit *Kunde* verknüpft. Es ist nun möglich, die Verknüpfung *Kontakte* zu *Kunde* oder auch die Verknüpfung *Kontakte* zu *Aufträge* für die Rechnung zu nutzen. Der Nachteil der ersten Variante ist jedoch, dass man so den Kontakt nicht direkt auswählen kann, sondern nur den Kunden, wobei immer nur sein erster Kontakt in der Rechnung erscheinen würde.

Den Bericht erzeugen wir wie gewohnt als *Administrator* unter dem Menüpunkt *Berichte*. Als Ursprungstabelle wählen wir die Tabelle *Aufträge* mit dem Namen „Rechnung“. Im nun folgenden Berichtseditor sehen wir in der Werkzeugleiste oben den gewählten Berichtsnamen und seine zugehörige Tabelle. Die zugehörige Tabelle ist immer die Tabelle, aus der sich der Bericht seine Inhalte holt. Ist die Tabelle mit anderen Tabellen verknüpft, kann der Bericht auch Inhalte aus diesen Tabellen holen. Dabei wird immer nur der erste Datensatz aus einer Verknüpfung abgefragt. Sollen mehrere Datensätze angezeigt werden, muss ein Tabellenelement genutzt werden und das Häkchen *Liste* im Kontextmenü des Elements aktiviert werden.

Wird das Häkchen LISTE außerhalb einer Tabelle ausgewählt, werden – falls vorhanden – alle Einträge angezeigt, die mit dem jeweiligen Datensatz verknüpft sind. Mithilfe des Kontextmenüpunkts SEPERATOR kann das Trennsymbol definiert werden. Beispielsweise ein *;* oder eine neue Zeile *\n*.

Als Erstes erstellen wir nun unseren Briefkopf. Dazu fügen wir ein Textelement hinzu und positionieren es an die Stelle, wo es am besten im Briefkuvert zu sehen ist. Der Inhalt ist der eigene Absender. Direkt darunter erstellen wir eine Tabelle mit 5 Zeilen und einer Spalte. Die Positionierung der Tabelle sollte ebenfalls dem Fenster des Briefkuverts entsprechen. In die einzelnen Zeilen fügen wir nacheinander die Adressdaten des Kunden ein. Dabei können mehrere Werte wie *PLZ* und *ORT* in eine Zeile eingefügt werden. Sie erscheinen durch Leerzeichen getrennt. Da wir den Ansprechpartner im Datenmodell gesondert verknüpft haben, können wir den Ansprechpartner unabhängig vom Kunden eintragen. Alle Daten, die mit dem Auftrag verknüpft sind, können in der Werkzeugleiste über den Punkt DATENINHALTE ausgewählt werden, indem man die verknüpfte Tabelle über das Plus-Symbol aufklappt.

Tabellen können ohne schlechtes Gewissen für einfache Aufgaben wie die Briefanrede benutzt werden. Sie ermöglichen es, mehrere Felder hintereinander zu setzen, ohne die genaue Länge des vorderen Elements zu kennen. Außerdem werden alle Elemente, die zu dieser Tabelle gehören, gemeinsam verschoben. Die Formatierung aller Elemente in einer Zelle kann nur über das erste Element eingestellt werden.

Als Nächstes fügen wir auf gleiche Weise die Auftragsnummer, die Briefanrede sowie Ort und Datum der Rechnung hinzu. Die Rechnungsbeschreibung ist ein längerer Text, der als Einzelelement unter die Briefanrede positioniert wird. Da es sich in diesem Fall um ein *Long*- oder *Blob*-Datenfeld handelt, hat das Häkchen HTML eine besondere Bewandtnis. Ist der Inhalt des Datenbankfelds *HTML*, wird der Bericht versuchen, den Inhalt zu interpretieren und in PDF zu konvertieren. Interessant wird dies bei Nutzung des WYSIWYG-Editors in den LIMBAS-Formularen. So können Teile direkt aus der Datenbank heraus in dem Bericht formatiert werden.

Jetzt wird es Zeit, die Positionen einzufügen. Dazu legen wir wieder eine Tabelle an. Diesmal allerdings mit 6 Zeilen und 5 Spalten. Da wir nicht wissen, wie viel Platz die Rechnungsbeschreibung benötigt, ist es wichtig, die Tabelle mit relativer Positionierung zum Element *Rechnungsbeschreibung* bzw. zu seiner *ID* zu versehen. Das geschieht mit dem Kontextmenüpunkt RELATIV. Dort wählen wir einfach die Elementnummer des Elements aus, zu dem die Positionierung relativ sein soll und bestätigen mit ÜBERNEHMEN. In die erste Zeile der Tabelle fügen wir nun die Beschriftungen der Spalten als Textelement hinzu. Die zweite Zeile beinhaltet dann die jeweiligen Datenelemente wie *Name*, *Beschreibung* oder *Einheit*. In die fünfte Spalte legen wir aus der Verknüpfung *Positionen* die Datenelemente *Betrag* und *Betrag Brutto*, wobei für *Betrag Brutto* der Kontextmenüpunkt VERSTECKT auf IMMER gesetzt wird. Damit wird zwar das Feld *Betrag Brutto* nicht angezeigt, kann aber für spätere Berechnungen in eigenen Formeln genutzt und ausgelesen werden. Wichtig zu beachten ist, dass alle Elemente der zweiten Zeile das Häkchen LISTE aktiviert haben, ansonsten wird nur das erste Verknüpfungsergebnis angezeigt. Zuständig für alle individuellen Inhalte, die erst im Bericht erzeugt werden sollen, ist das *Formel*-Element, das in der Werkzeugleiste ausgewählt werden kann. So können wir in Zeile vier die anfallende Mehrwertsteuer anzeigen. Dazu wird bei einheitlicher Mehrwertsteuer der Betrag direkt aus der Summe der Nettobeträge berechnet, oder wie in unserem Fall die Summe der einzelnen (versteckten) Bruttobeträge als Berechnungsgrundlage genutzt. Nachdem wir ein Formelelement in das entsprechende Tabellenfeld eingefügt haben, editieren wir den Inhalt über den Kontextmenüpunkt EDIT. Eine Formel wird in PHP-Code geschrieben – es kann beliebiger Code sowie auch ein Funktionsaufruf eingetragen werden. Die einzige Regel ist, dass das Ergebnis, was dann auch angezeigt werden soll, in die Variable *$arg_result* übertragen wird. Die folgende Zeile zeigt eine einfache Formel, die „hallo Limbas“ ausgeben soll: *$arg_result = "hallo Limbas"*.

Um nun die Mehrwertsteuer zu berechnen (folgender Quellcode), benötigen wir noch die Brutto- und Nettobeträge aller Positionen. Diese sind über das Array *$glob_el* verfügbar. Dabei ist der erste Schlüssel die Element-ID, die bei Klick auf das Element im Kontextmenü angezeigt wird. Der zweite Schlüssel ist die Zeilennummer der Tabelle, die immer bei 1 im Tabellenkopf beginnt:

```
$arg_result = number_format(array_calculate($glob_
el[57]) - array_calculate($glob_el[45]),2,“,“,““).“ €“;
```

Das Array für den Bruttobetrag könnte demnach so aussehen:

```
$glob_el[57][2] = 15
$glob_el[57][3] = 4,5
$glob_el[57][4] = 12,34
```

Unser Beispiel nutzt noch eine kleine Funktion (Listing 1), die die Array-Werte addiert. Sie kann als Erweiterung in das *EXTENSION*-Verzeichnis von LIMBAS erstellt werden oder in einer weiteren Formel. Diese muss allerdings vor der Berechnungsformel ausgeführt werden, was durch den Z-Index steuerbar ist. Analog dazu erzeugen wir den Rechnungsbetrag: *$arg_result = number_format(array_calculate($glob_el[57]),2,“,“,““).“ €“;*.

Möchte man eine Zeile ausblenden, falls ein Artikel keinen Preis besitzt oder gar negativ ist, kann man die folgenden Zeilen verwenden. Dabei ist die Variable *$ROWNR* immer die aktuelle Zeile, in der man sich befindet, und das Ergebnis *#HIDEROW* bewirkt das Verstecken der aktuellen Zeile. Will man auf Ele-

Listing 1

Addition der Array-Werte

```
function array_calculate($el){
 if(is_array($el)){
  foreach ($el as $key => $value){
   $val += doubleval(str_replace(“,“,“.“,$value));
  }
  return $val;
 }
 return 0;
}
```

mente vor oder nach der aktuellen Zeile zugreifen, genügt ein *$ROWNR+1* oder *-1*:

```
if (!$glob_rowel[57][$ROWNR]){
 $arg_result="#HIDEROW";
}
```

Mit einem abschließenden Satz am Fuß des Berichts und dem Setzen des Dokumentfußes ist die Berichtsvorlage vollständig. Wie wir sehen, ist es mit Nutzung der Formelelemente möglich, auch komplexeste Berichte zu erstellen. In der Praxis holen sich die Formeln ihre Werte aus größeren Funktionen, die für die Berechnung zuständig sind. Auch das Zusammensetzen mehrerer Berichte ist möglich. Dazu können die Berichte einzeln erstellt und über ein eigenes Erweiterungsskript nach Wunsch zusammengefügt werden. Auch Inhaltsverzeichnisse sind somit umsetzbar. Einrichten kann man so eine Erweiterung in der Berichtsauswahl über den Punkt BERICHTERWEITERUNG. Existiert für einen Bericht eine Erweiterung, wird diese anstatt des normalen Berichts ausgeführt (Listing 2).

Um Erweiterungen, also eigenen Code in LIMBAS zu integrieren, gibt es verschiedene Möglichkeiten. In unserem Fall ist es das Einfachste, eine Datei namens *ext_report.inc* in dem Verzeichnis */EXTENSIONS/* zu erstellen, in dem eigene Funktionen oder Klassen zu finden sind. Es können zur Übersicht mehrere Ordner und Unterordner erstellt und dort jeweils wieder *ext_*.inc*-Dateien angelegt werden. Dieses System funktioniert übrigens für alle wichtigen Programmteile von LIMBAS. Es muss lediglich das * mit dem eigentlichen Dateinamen ersetzt werden.

Eine Alternative zur Erstellung von Berichten bietet das Berichtsmodul für OpenOffice.org Writer.

Eine weitere Alternative, Berichte mit LIMBAS zu generieren, ist das neue Berichtsmodul für *OpenOffice.org Writer*. Mit dieser Erweiterung lässt sich auf einfachste Weise ein OpenOffice.org-Dokument aus einer schon vorhandenen PDF-Vorlage erstellen. Dazu wird der Bericht zuerst als XML geschrieben, womit dann letztendlich die Vorlage ergänzt wird. Das Schöne daran ist, dass aus der gleichen Berichtsvorlage entweder ein PDF-, ein XML- oder ein OpenOffice.org-Writer-Dokument erstellt werden kann. Um einen OpenOffice.org-Bericht zu generieren, benötigen wir eine Vorlage, die selbst ein OpenOffice.org-Dokument ist. Wir definieren lediglich die Platzhalter, die später ersetzt werden sollen (*odt*-Template, Kasten: „Auszug aus einer *.odt*-Template-Datei"). Zur Übersicht können wir uns den Bericht als XML ausgeben lassen und die Namen mit den Platzhaltern vergleichen. Danach speichern wir es im LIMBAS-Dateimanager in einem beliebigen Verzeichnis und wählen es im Admin-Bereich der Berichtsübersicht als Office-Vorlage aus. In der LIMBAS-Demoinstallation befindet sich eine solche Vorlage für das Produktdatenblatt. Nun können wir nach einem *reset* aus einem beliebigen Produkt den Bericht *Produktdatenblatt* erzeugen und über das Auswahlfeld FORMAT das Ausgabeformat wählen. Für die nächste Version ist es geplant, Bilder direkt aus dem LIMBAS DMS zu integrieren.

Im Allgemeinen ist ein einfacher Bericht schnell und einfach erstellt. Auch komplexere Berichte sind durch die Erweiterungsfunktionalitäten möglich. Obwohl LIMBAS völlig auf Browser-Plug-ins verzichtet und nur mit JavaScript arbeitet, ist die Handhabung des Berichtseditors durchaus komfortabel. Die Entwickler sehen die Vorteile einer reinen Browserapplikation als vorrangig gegenüber gängigen Desktopapplikationen. Dazu kommt, dass sich die Handhabung mit den neueren Browsern stetig verbessert. Da nun auch ein OpenOffice.org-Bericht zu Verfügung steht, dessen Layout völlig unabhängig erzeugt werden kann, gibt es neben dem LIMBAS-PDF-Bericht diese Alternative.

Auszug aus einer .odt-Template-Datei

${zeit_von} - ${zeit_bis}
Land: ${land}
Ort: ${ort}
PLZ: ${plz}

${positionen:name}	${positionen:preis}

Hier die zugehörige XML-Ausgabe:

```
<report name="Chronik">
<item id="1">
<text name="zeit_von">17.10.1977</text>
<text name="zeit_bis">22.10.1977</text>
<text name="land">Schweiz</text>
<text name="ort"/>
<text name="plz">12356</text>
</item>
</report>
```

Listing 2

Erweiterung eines Berichts

```
$gtabid = 1;  #Tabellen ID
$ID = 1;   # Datensatz ID
$report_output = "pdf"; # Ausgabeformat
$report_id = 1;  # Berichts ID
$grouplist = get_report_group($report_id,array());
$generated_file_path = generate_
pdf($grouplist,$ID,$report_output);
$filelist[] = $generated_file_path;
if (count($filelist) > 1){
 $mergefile = merge_report($filelist,'0');
 view_report($mergefile);
} elseif ($generated_file_path){
 view_report($generated_file_path);
}
```

Axel Westhagen *(axel.westhagen@limbas.de) rief 1998 das LIMBAS-Projekt ins Leben, was 2006 die Gründung der gleichnamigen Firma zur Folge hatte. Die LIMBAS GmbH widmet sich ganz der Entwicklung des LIMBAS-Frameworks und dessen Support.*

Links & Literatur

[1] Westhagen, Axel: „PDF-Erstellung leicht gemacht", in: Entwickler Magazin 6.09, S. 76 ff.