Ein Webformular entwickeln = drei Manntage ???
Budget sparen mit PL/SQL und XML-Formularen

Martin Friemel / Martin Kubitza
Enterprise Web AG
Duisburg

Schlüsselworte

Intranet, Web-Formularanwendung, XML, PL/SQL Softwareentwicklung, Oracle RDBMS Standard Edition, Apache Web-Server, WebAG Automat, www.webag.com

Zusammenfassung

Der Vortrag zeigt, wie mit Hilfe einer XML-basierten Formular-Engine kostengünstig robuste und leicht zu wartende Web-Formularanwendungen realisiert werden. Die Vorteile der datenmodellunabhängigen XML-Speicherung von Formulardefinitionen und Formulareingaben werden anhand einiger Projektbeispiele und PL/SQL-Sourcecode-Auszügen beschrieben.

Inhalt

- Einleitung
- Konzept Formularsystem
    - Grundlage: Formulardefinitionen in XML
    - Verarbeitung: Formular-Treiber
    - Speicherung: XML-Formular-Container
- Formularsystem PL/SQL-API für Entwickler
- Anwendungsbeispiele aus der Praxis
- Fazit

Einleitung

Leider wird in vielen Web-Projekten die eingesetzte Softwareentwicklungs-Architektur und die Ablaufumgebung der Webmasken nicht nach den Kriterien *kostengünstige Softwareentwicklung, robuster Betrieb* und *leichte Wartbarkeit* ausgewählt.

Statt dessen wirken die Projekte gerade bei Web-Anwendungen oftmals wie ein Schaulaufen der neuesten Spezialitäten im Rennen der verschiedenen Applikationsserver. Selbst einfachste Web-Formulare entpuppen sich als hochkomplizierte Gebilde auf dem Weg durch die einzelnen Schichten vom Webserver durch Java-Serverpages, Enterprise Java Beans, Datenbankobjekte, JDBC-Abfragen bis hinunter in die zugrundeliegende Datenbank.

Trotzdem erweist sich allein die Realisierung der Darstellungsschicht für die Abwicklung der Formulare auf dem Browser-Bildschirm als Albtraum, denn es werden Experten benötigt in den Disziplinen Java/JSP, Javascript, Cascading Stylesheets und HTML. Glücklich, wer das in einer Person vereint!

Das benötigte Know-How der beteiligten Entwickler ist beachtlich, doch die kurze Halbwertzeit der momentan „modernen“ Architektur bei den einzelnen Applikationsservern spricht gegen die flächendeckende Schulung der IT-Mitarbeiter.

Zu allem Ungemach leiden die so entstandenen Web-Anwendungen häufig unter einer miserabelen Performance.

Als Hersteller eines Web-Contentmanagementsystems (Web-CMS) werden wir oft mit Anforderungen für die Umsetzung konkreter Web-Formularanwendungen konfrontiert. Wir verwenden ein Formular-Konzept, das die Techniken der Oracle-Datenbank nutzt. Das Konzept besteht aus drei Komponenten, die im Folgenden detailliert beschrieben werden:

- Formulare werden im XML-Format definiert und in der Datenbank gespeichert. Ein Web-basierter Formular-Editor erleichtert das Erstellen einer Formular-Definition.

- Eine PL/SQL-Formular-Engine stellt die Formulare auf dem Brower dar. Sie steuert die Benutzereingaben, die Speicherung und Weiterverarbeitung der Eingabewerte. Dazu wertet die Engine die XML-Formulardefinitionen aus.

- Die Eingaben der Formular-Benutzer werden ebenfalls im XML-Format in der Datenbank gespeichert, damit sie kein eigenes Datenmodell benötigen und trotzdem mit Hilfe der Oracle XML-Parser leicht von ggfs. bestehender Backend-Software weiterverarbeitet werden können.

Für die Umsetzung werden eine Oracle RDBMS Standard Edition, der darin enthaltene Apache Web-Server sowie ggf. HTML-, XML- und PL/SQL-Grundkenntnisse zum Verständnis der Anwendungen benötigt.

Konzept Formularsystem

Die drei Komponenten des Formular-Konzepts werden nachfolgend beschrieben und ihre Vorteile erläutert. Als praktisches Beispiel wird ein einfaches Kontaktformular gezeigt. Im Web sollen die Benutzer ihren Namen und die E-Mail-Adresse eingeben können, um damit einen Newsletter zu abonnieren.

Dieses Kontaktformular besteht aus zwei Textfeldern, einem Kontrollkästchen und einem mehrzeiligen Texteingabebereich:

Abb. 1: Formular mit verschiedenen Feldkomponenten

Grundlage: Formulardefinitionen in XML

Unsere XML-Beschreibung speichert für dieses Formularbeispiel folgende Syntax:

```
<wt_FORM NAME="p_81_form">

   <wt_TEXT NAME="p_name_text" SIZE="40" ID="82"
            LABEL="Vorname, Name" REQUIRED="FALSE"/>

   <wt_TEXT NAME="p_email_text" SIZE="40" ID="86"
            LABEL="email-Adresse" REQUIRED="FALSE"/>

   <wt_CHECK NAME="p_newsletter_check" VALUE="default"
            CHECKED="FALSE" ID="85"
            LABEL="Ich möchte den Newsletter erhalten"/>

   <wt_TEXTAREA NAME="p_bemerkung_textarea" ROWS="4"
```

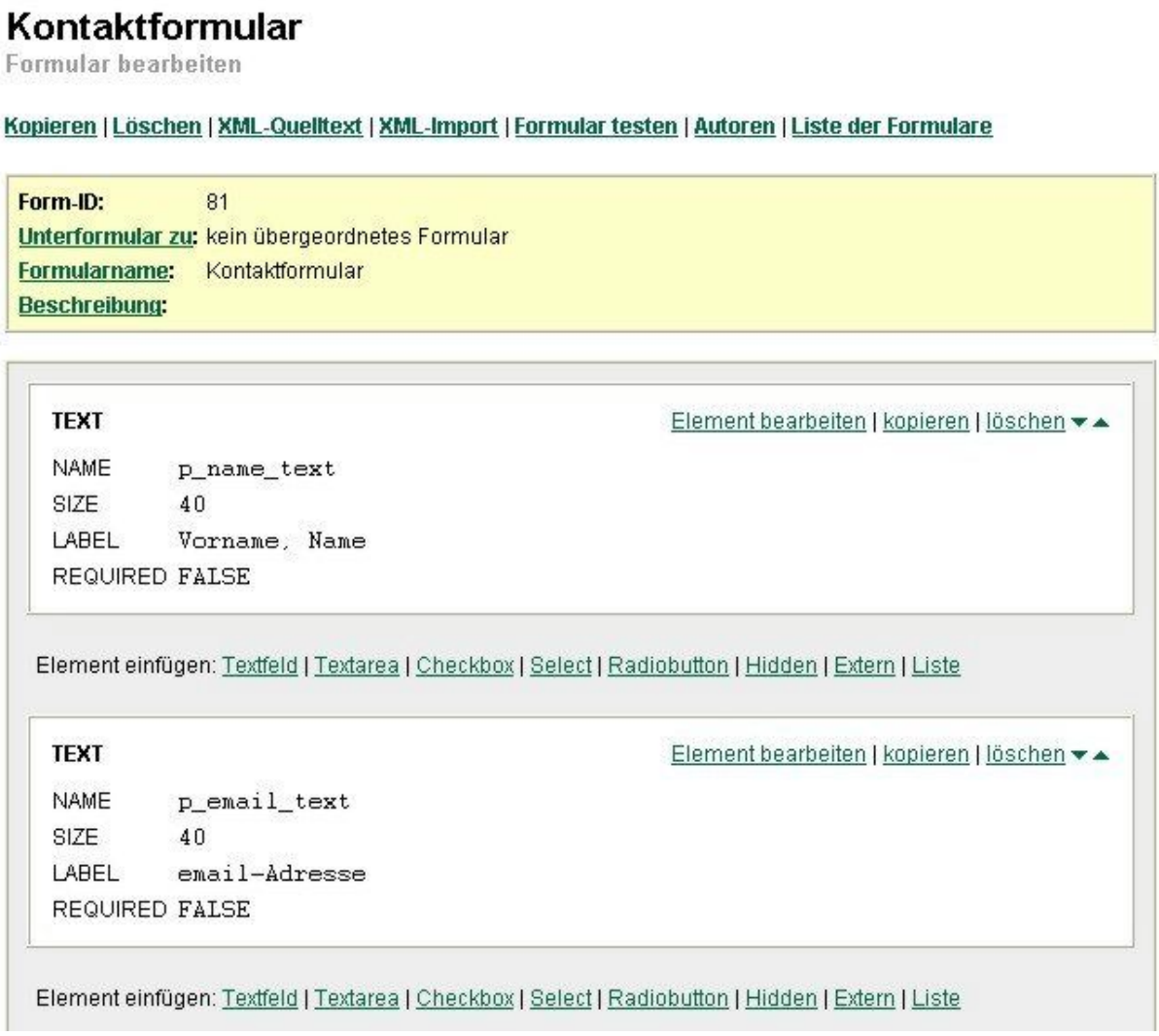
```
            COLS="68" ID="84" LABEL="Bemerkung" REQUIRED="FALSE"/>

</wt_FORM>
```

Abb. 2: XML-Syntax des Kontaktformulars

Es kann natürlich nicht verlangt werden, dass diese Syntax von Hand eingetippt werden. Bei umfangreichen Formularen kann der XML-Code nämlich schnell unübersichtlich werden.

Statt dessen geschieht die Eingabe deklarativ mit einem Web-basierten **Formulareditor**, mit dem die einzelnen Eingabefeld-Typen mit allen erlaubten Attributen per Mausklick in das Formular eingefügt werden können. Der Formulareditor speichert das korrekte XML-Format in die Datenbank. Hier ein Screenshot des Formulareditors für das Beispiel-Kontaktformular:



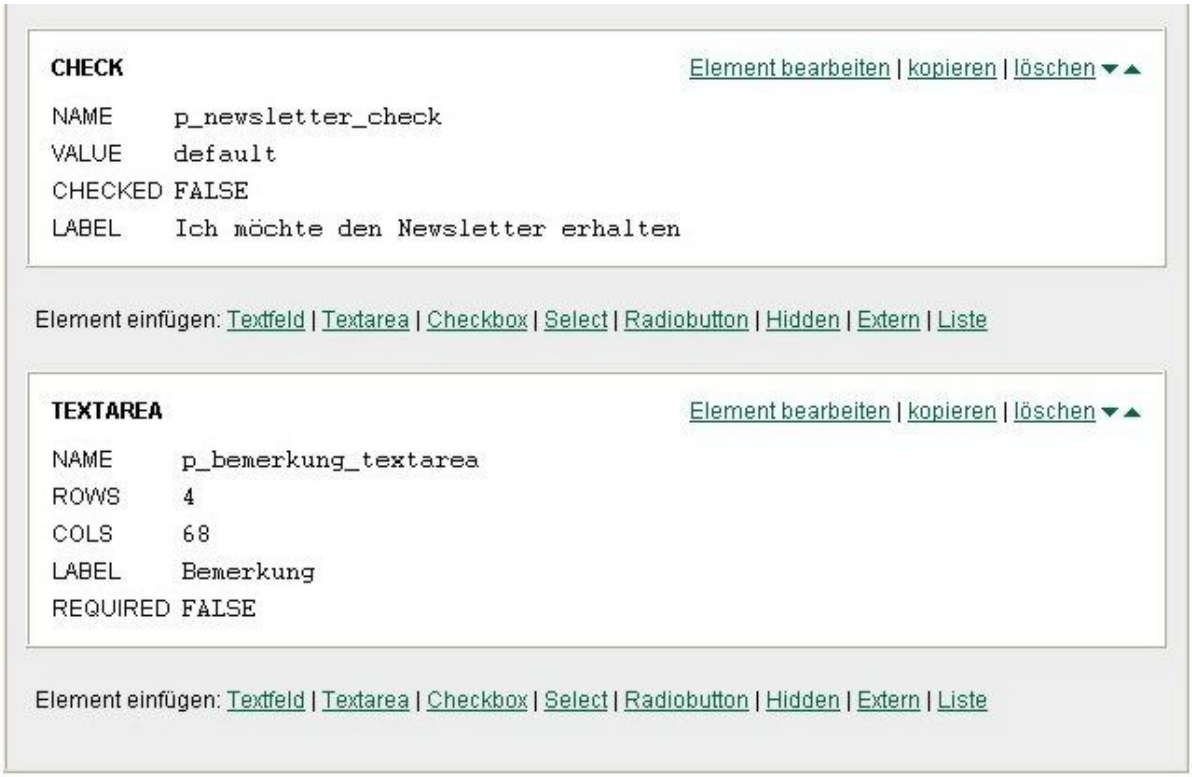


Abb. 3: Formular-Editor

Der Editor ermöglicht die Definition der verschiedenen Eingabefeld-Typen:

- Einzeiliges Textfeld
- Mehrzeiliger Textbereich (*Textarea*)
- Auswahlfeld (*Dropdown*)
- Kontrollkästchen (*Checkbox*)
- Optionsfeld (*Radiobutton*)
- verstecktes Feld (*Hidden*)
- selbstentwickeltes (*Custom*-) Feld für Eingabefelder mit eigener Eingabe oder Auswahl-Logik
- Gruppierung der o.a. Felder zu einer Liste, damit untereinander mehrere Eingaben der Felder ermöglicht werden können.

Außerdem können wichtige Feldattribute von der Beschränkung der Eingabelänge bis hin zu selbst programmierten Validierungen der Eingaben eingetragen werden. Alle diese Informationen zu dem Formular werden in der XML-Formulardefinition gespeichert.

Enge Anbindung an die Datenbank

Dropdown-Formularfelder können untereinander in Abhängigkeit gebracht werden, d.h. die Auswahlmöglichkeiten eines Dropdown-Formularfeldes B können in Abhängigkeit der Auswahl eines Dropdown-Formularfeldes A aufgebaut werden.

Ein Anwendungsbeispiel dafür ist die Online-Bewerbung um das Feodor-Lynen Programm bei der Alexander von Humboldt-Stiftung:

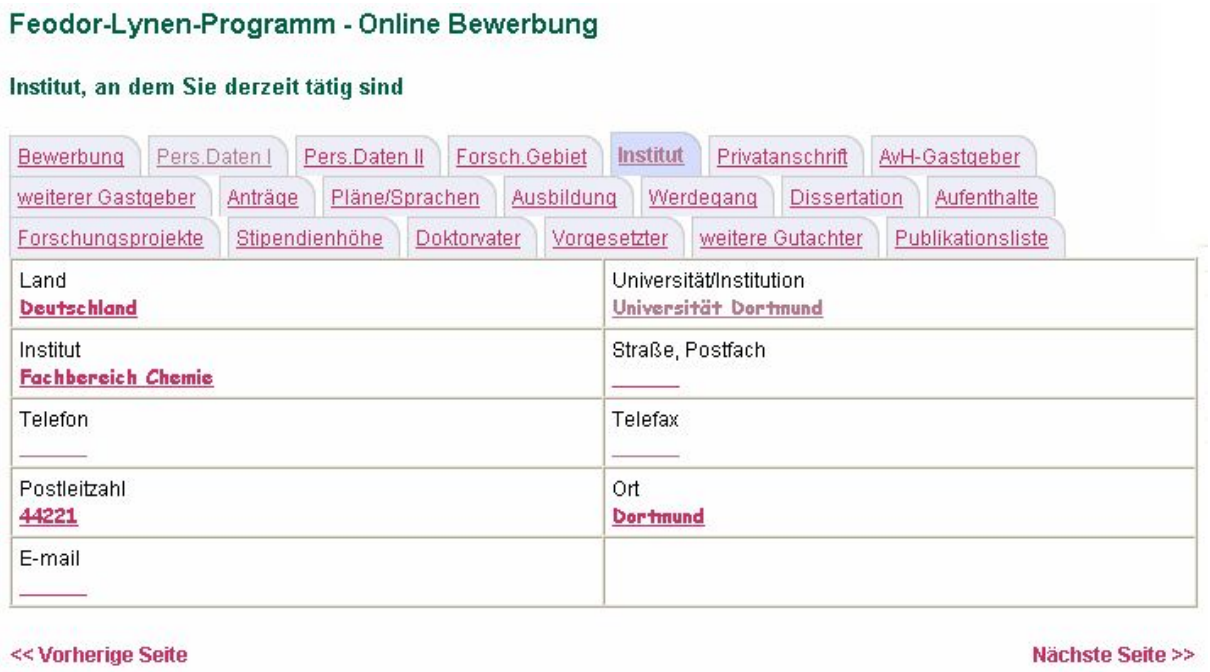


Abb. 4: Formular mit abhängigen Dropdown-Feldern

Die Dropdown-Felder *Land*, *Universität/Institution* und *Institut* hängen voneinander ab. Ihre Wertelisten werden aus dynamischen SELECTs zusammengestellt. Dafür wird in den abhängigen SELECTs die Feldbezeichnung als ${Bezeichnung} des übergeordneten Formularfeldes eingefügt. Diese SELECTs zur Befüllung der Wertelisten werden ebenfalls in der XML-Formulardefinition gespeichert.

LAND_select:

```
SELECT name, value
FROM land_v
ORDER BY name
```

UNIVERSITAET_select:

```
SELECT name, value
FROM universitaet_v
WHERE land = '${LAND_select}'
ORDER BY name
```

INSTITUT_select

```
SELECT instituts_bezeichnung
NAME, instituts_id VALUE
FROM institut_v
WHERE uni_id =
      '${UNIVERSITAET_select}'
ORDER BY NAME
```

Abb. 5: Wertelisten-SELECTs für Abhängigkeiten

Das Beispiel zeigt außerdem, dass Formulare zu einer Formularmappe zusammengebunden werden können. Über Karteireiter wird die Navigation zwischen den Einzelformularen realisiert. Dadurch ist die Darstellung am Bildschirm viel übersichtlicher als bei einem sehr großen, einseitigen Bildschirmformular.

Validierungen

Formularfelder, in denen Eingaben zwingend notwendig sind, können als *Pflichtfelder* definiert werden. Beim Abschicken des Formulars werden fehlende Eingaben eingefordert:

Abb. 6: Pflichtfelder

Eingabevalidierungen überprüfen beim Abschicken des Formulars Felder auf syntaktische Korrektheit. Zur Verfügung stehen Standardvalidierungen für Zahlen- und Datumsformate, eine Erweiterung auf beliebige Validierungen durch eigene PL/SQL-Prozeduren ist möglich.

An jedes Eingabefeld können *Ereignisse* gehängt werden. Dadurch kann der Softwareentwickler mit einer eigenen PL/SQL-Prozedur eingreifen, wenn z.B. in ein bestimmtes Eingabefeld ein Wert eingegeben wird (Ereignis *ON_CHANGE*). Es könnte z.B. sinnvoll sein, in Abhängigkeit von dieser Eingabe ein anderes Feld neu zu belegen. Herkömmliche Webformulare müssen dazu Javascript verwenden und können dabei nicht auf die Datenbank zugreifen.

Nach der Besprechung der reichhaltigen Informationen, die in der XML-Formulardefinition gespeichert werden, drängen sich folgende Fragen auf:

- Wer wertet die XML-Formulardefinition aus und bringt das Formular auf den Browserbildschirm?

- Wie sieht das Formular dann auf meiner Webseite aus?

- Was geschieht nach der Eingabe mit den Daten, die in die Eingabefelder eingetragen wurden?

Die Beantwortung dieser Fragen leitet über zur zweiten Komponente des Formularsystems, dem Formulartreiber. Der Formulartreiber ist die Engine, die die Formulare ausführt, die Eingaben entgegennimmt und die Weiterverarbeitung auslöst.

Verarbeitung - Formulartreiber

Für die Ausführung des XML-Formulars ist der Formulartreiber zuständig. Alle Einstellungen und Definitionen, die zum Betrieb des Formulars notwendig sind, werden hier gespeichert. Dazu gehören Informationen zum Webdesign, zur Weiterverarbeitung der Eingabedaten oder auch zu Autorisierungsfragen. Die Trennung dieser Einstellungen von den Formulardefinitionen ist sinnvoll, weil ein Formular dadurch in unterschiedlichen Umgebungen oder Anwendungen verwendet werden kann.

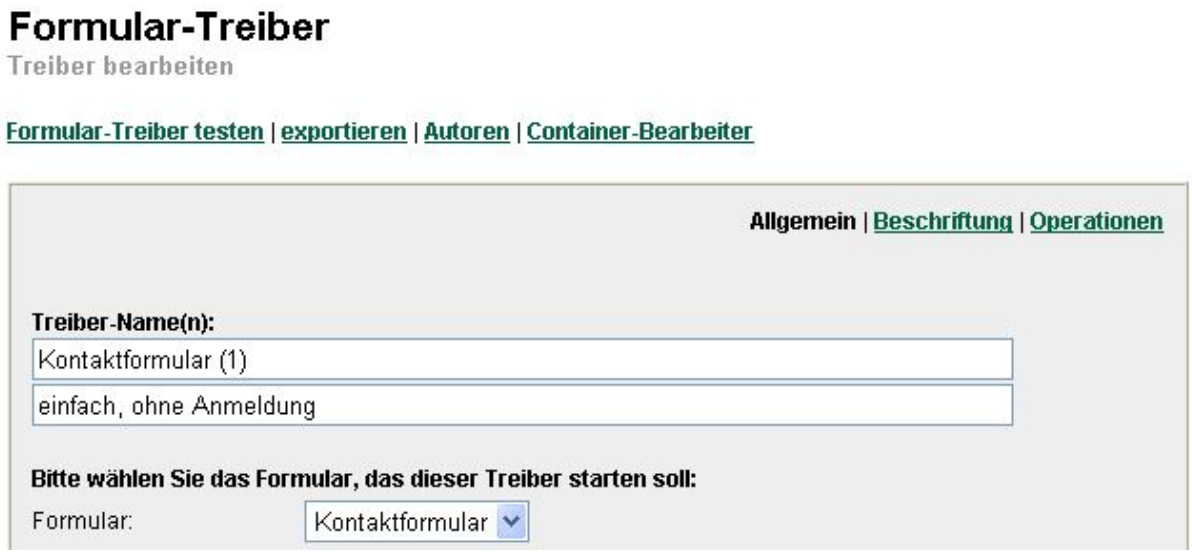


Abb. 7: Formular-Treiber

Für die Darstellung von Formularmasken wird ein Weblayout genommen. Wir verwenden dazu Funktionen unseres Web-Contentmanagementsystems WebAG Automat. Dadurch ist gewährleistet, dass die Formulare nahtlos in das Design der übrigen Webseiten eingepasst werden.

Feldbeschriftungen können unterschiedlich positioniert werden. Das Aussehen der Eingabefelder kann mit Stylesheets angepasst werden.

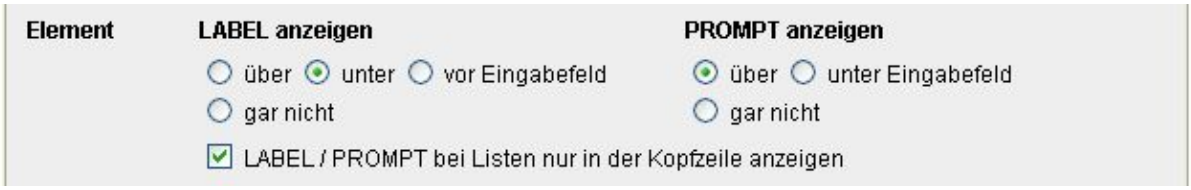


Abb. 8: Beschriftung (Formularfelder)

Die üblichen Bestätigungsabfragen – z.B. vor dem Abbrechen, Abspeichern etc. - können einzeln als Text und/oder als Grafik vorgegeben werden, mit einer Sicherheitsabfrage versehen oder ein-/ausgeblendet werden.

Abb. 9: Beschriftung (Formular-Buttons)

Autorisierung

Die Formularanwendung kann anonym oder authentifiziert, d.h. nur nach Anmeldung, laufen. Authentifizierte Formulare können vom angemeldeten Benutzer auch nachträglich bis zum Abschicken beliebig oft aufgerufen und weiter bearbeitet werden. Dieser Modus eignet sich z.B. für Formularmappen mit vielen Eingabefeldern.

HANDLER - PL/SQL-Operationen

Für bestimmte Situationen beim Betrieb des Formulars sind über Handler verschiedene Operationen in Form eigener PL/SQL-Prozeduren möglich. Diese können zum Zeitpunkt der Initialisierung (Erstaufruf), beim Abspeichern oder Abschicken sowie beim Abbruch oder Löschen ausgeführt werden.

- Der *INIT HANDLER* wird bei der Initialisierung des Formulars ausgeführt und kann z.B. zur Vorbelegung von Feldern benutzt werden.

- Der *APPLY HANDLER* wird beim Zwischenspeichern des Formulars ausgeführt.

- Der *SUBMIT HANDLER* wird beim Abschicken des Formulars ausgeführt und kann z.B. einen Workflow zur Verarbeitung der Eingabedaten anstoßen.

- Der *CANCEL HANDLER* wird beim Abbruch einer Eingabe ausgeführt.

- Der *DELETE HANDLER* wird beim Löschen eines Formulars ausgeführt.

Mit den Treiber-Informationen kann die PL/SQL Formularengine des Formularsystems die Formulare über die Oracle modPLSQL-Schnittstelle ausführen. Die WebAG Automat-Designauswahl sorgt für die optische Einbindung in das Webdesign, die Formulardefiniton beschreibt die Eingabefelder und die Handler im Treiber rufen die PL/SQL-Routinen der Entwickler zur Weiterverarbeitung der Eingabedaten auf. Diese Handler sind übrigens die einzigen Komponenten, die von den Formular-Entwicklern von Hand programmiert werden müssen.

Die Standardausgabe der Formulare mit Layouts und Parametern aus dem zur Verarbeitung verwendeten Formular-Treiber führt in der Regel zu befriedigenden Ergebnissen. Noch mehr Möglichkeiten haben PL/SQL-Entwickler mit dem Einsatz der PL/SQL-API des Formularsystems bei eigenen Web-Formularanwendungen. Am Ende des Vortrags werden dazu einige Sourcecode-Beispiele gezeigt.

Speicherung: XML-Formular-Container

Formulareingaben werden zunächst im XML-Format in einem sogenannten Formular-Container gespeichert. Die Formulare sind damit unabhängig von bestehenden relationalen Datenmodellen. Eine spätere oder gleichzeitige Übernahme in ein vorhandenes relationales Datenmodell ist durch die Ausführung eines dafür geschriebenen *SUBMIT_HANDLER* möglich. Hierfür bietet eine PL/SQL-API Programmierern hilfreiche Aufrufe. Auch das Oracle XML Developer Kit (XDK) kann beim Parsen des XML-Containers eingesetzt werden.

Im Formular-Container wird die gesamte XML-Formulardefinition mit der Benutzereingabe gespeichert. Da die Ausgabe der Formulare auf Basis des Formular-Containers erfolgt, ist

auch bei späteren Änderungen der zugrunde liegenden Formulardefinition jederzeit ein Arbeiten auf dem alten Datenbestand möglich, da dieser von den Änderungen nicht betroffen ist.

Abb. 10: Formular-Container (Text-Ansicht)

```
<wt_FORM NAME="p_81_form">

   <wt_TEXT NAME="p_name_text" SIZE="40" ID="82"
            LABEL="Vorname, Name" REQUIRED="FALSE">
       Martin Kubitza
   </wt_TEXT>

   <wt_TEXT NAME="p_email_text" SIZE="40" ID="86"
            LABEL="email-Adresse" REQUIRED="FALSE">
       kubitza@gmx.net
   </wt_TEXT>

   <wt_CHECK NAME="p_newsletter_check" VALUE="default"
            CHECKED="FALSE" ID="85"
            LABEL="Ich möchte den Newsletter erhalten"/>

   <wt_TEXTAREA NAME="p_bemerkung_textarea" ROWS="4"
            COLS="68" ID="84" LABEL="Bemerkung" REQUIRED="FALSE"/>

</wt_FORM>
```

Abb. 11: Formular-Container (XML-Ansicht)

Formularsystem PL/SQL-API für Entwickler

PL/SQL-Entwickler können mit der PL/SQL-API ihre eigenen Formular-Anwendungen programmieren und dabei auf dieselben Prozeduren und Funktionen zurückgreifen, die das Formularsystem selbst in der Standardausgabe verwendet. Auf diese Weise lassen sich anspruchsvolle Formular-Anwendungen mit überschaubarem Aufwand realisieren.

Die folgenden Quelltext-Auszüge sollen die wenigen programmiertechnischen Handgriffe aufzeigen, die zur Entwicklung einer eigenen Formular-Anwendung notwendig sind.

Schrittweise zur eigenen Formular-Anwendung in PL/SQL

Zuerst wird auf der Basis eines Formular-Treibers (`i_form_driver_id`) ein Formular-Container mit dem jeweiligen Rohformular initialisiert und in der Datenbank angelegt. Der Container enthält die Formulardefinition und wird außerdem die Benutzereingaben im XML-Format speichern.

```
--------
-- initialisiere neues Formular
--------
wt_form_api.init_form_container (
   i_user_id              => wt.g_user.user_id,
   i_form_driver_id       => pub_form.form_driver_id,
   o_form_container_id    => l_form_container_id
);
```

Abb. 12: Initialisiere neues Formular

Der Formular-Container wird zur Verarbeitung über seine ID (`i_form_container_id`) in den Datentyp (`o_form_container_rec`) geholt.

```
--------
-- Hole Formular-Container
-------
wt_form_api.get_form_container (
   i_form_container_id       => l_form_container_id,
   o_form_container_rec      => l_form_container_rec
);
```

Abb. 13: Hole Formular-Container

Aus der XML-Formulardefinition (`i_xml_data`) im Formular-Container wird das Document Object Model (DOMDocument) aufgebaut, auf dem die nachfolgenden Operationen stattfinden.

```
--------
-- initialisiere XML-DOMDocument
--------
wt_form_api.init_xml (
   i_xml_data        => l_form_container_rec.xml_data,
   o_xml_domdocument => l_DOMDocument
);

-------
-- Root im DOMDocument finden
-------
l_RootElement := xmldom.getDocumentElement(l_DOMDocument);
l_RootNode    := xmldom.makeNode(l_RootElement);
```

Abb. 14: Iinitialisiere XML-DOMDocument

Das Formular wird geöffnet, indem die zur HTML-Darstellung benötigten Ausgaben gemacht werden.

```
----------
-- oeffne Formular
----------
htp.p (
   wt_form_api.form_open (
      i_form_id            => l_form_container_rec.form_id,
      i_form_driver_id     => l_form_container_rec.form_driver_id,
      i_form_container_id  => l_form_container_rec.form_container_id
   )
);
```

Abb. 15: Öffne Formular

Die einzelnen Formularfelder können über ihre interne Bezeichnung (`i_element_name`) ausgegeben werden.

```
--------
-- Formularfelder anzeigen
--------
HTP.p ('<table border="0" cellpadding="3" cellspacing="3">');
HTP.p ('<tr>');
HTP.p (   '<td>'
    ||
       --------
       -- VORNAME (Textfeld)
       --------
       wt_form_api.form_element (i_element_name => 'p_vorname_text',
                                 i_DOMDocument  => l_DOMDocument
                                                  )
    || HTF.br
       --------
       -- VORNAME (Label-Attribut)
       --------
    || wt_form_api.form_attribut (i_element_name => 'p_vorname_text',
                                  i_element_attr => '@LABEL',
                                  i_DOMDocument  => l_DOMDocument
                                 )
   );
HTP.p ('</td>');
HTP.p
   (   '<td>'
    ||
       --------
       -- NAME (Textfeld)
       --------
       wt_form_api.form_element (i_element_name => 'p_name_text',
                                 i_DOMDocument  => l_DOMDocument
                               )
    || HTF.br
       --------
       -- NAME (Label-Attribut)
       --------
    || wt_form_api.form_attribut (i_element_name => 'p_name_text',
                                  i_element_attr => '@LABEL',
                                  i_DOMDocument  => l_DOMDocument
                                 )
   );
HTP.p ('</td>');
HTP.p ('</tr>');
HTP.p ('</table>');
```

Abb. 16: Formularfelder anzeigen

Buttons zur Formularsteuerung – wie hier „Abschicken“ – werden wie im Formular-Treiber eingestellt in HTML ausgegeben.

```
--------
```

```
-- "Abschicken"-Button
--------
HTP.p ( wt_form_api.form_submit_button
           (i_form_container_id => l_form_container_rec.form_container_id)
      );
```

Abb. 17: „Abschicken“-Button

Zuletzt wird das nach der Initialisierung mit HTML-Code geöffnete Formular wieder geschlossen.

```
--------
-- schliesse Formular
--------
htp.p( wt_form_api.form_close );
```

Abb. 18: Schließe Formular

Die vorgestellten Schritte werden in eine PL/SQL-Procedure eingebettet, die folgende Struktur aufweist:

```
--------
-- Profil aus einem Formular-Container bearbeiten
--------
PROCEDURE contact_edit (
   p_form_container_id  IN NUMBER DEFAULT NULL
)
IS
   l_DOMDocument        xmldom.DOMDocument;
   l_RootNode           xmldom.DOMNode;
   l_RootElement        xmldom.DOMElement;

   l_form_container_id  NUMBER := p_form_container_id;
   l_form_container_rec wt_form_container%ROWTYPE;

BEGIN

          /* Formular-Aufrufe aus der PL/SQL-API */

END; -- contact_edit
```

Abb. 19: PL/SQL-Procedure

Die Procedure kommt außer der ID des Formular-Containers (`p_form_container_id`) ohne weitere Angaben aus. Nur die Positionierung der Formular-Elemente des zugrunde liegenden Formulars ist im Quelltext einer ggf. vorher per HTML-Editor gelayouteten Seite noch notwendig. Die Felder selbst werden vom Formularsystem-API dargestellt.

Hier liegt eine enorme Erleicherung in der Entwicklung: Über die korrekte Parameterübergabe

der eingebenden Procedure (z.B. `contact_edit`) an eine verarbeitende Procedure müssen sich keine Gedanken gemacht werden, da die Verarbeitung (und somit auch die korrekte Parametrisierung der verwendeten Formularsystem-Procedure) über das Formularsystem erfolgt.

Anwendungsbeispiele aus der Praxis

Der DOAG-Vortrag wird an dieser Stelle um die Live-Präsentation eines aktuellen Projektes ergänzt, bei dem die hier beschriebenen Techniken eingsetzt werden.

Fazit

Das „Konzept Formularsystem“ ist unser Ansatz, um mit geringem Aufwand performante und robuste Formularanwendung zu entwickeln und zu betreiben. Die dabei eingesetzte Basis-Technologie wird mit der Oracle RDBMS Standard Edition ab Version 9.2i geliefert.

Mit unserem Formularsystem, das Bestandteil des WebAG Automat Autorensystems ist, lassen sich Formularanwendungen in kurzer Zeit und ohne großen Aufwand und Programmierkenntnisse umsetzen (Beispiel: Kontaktformular).

Wer - auch nur geringe – Kenntnisse in HTML, XML, SQL und PL/SQL hat, kann mit der integrierten PL/SQL-API des Formularsystems statt der Standardausgabe eigene Formularanwendungen entwickeln. Die Wartbarkeit von PL/SQL-Software und gerade unserer Formularanwendungen hat sich in vielen Oracle-Projekten bewährt.

Die XML-Speicherung der Formulardefinitionen und die zugehörige Formular-Engine machen die Web-Entwicklung einerseits unabhängig von zugrundeliegenden Datenmodellen, erlauben jedoch trotzdem eine leichte Befüllung der Formular-Wertelisten an bestehende Tabellen oder Views.

Der Weg der Formulareingaben über den Verarbeitungsworkflow in die Unternehmensdatenbank wird flexibel über die Handler-Technolgie gemäß ihrer speziellen Projektanforderungen realisiert. Programmieraufwand entsteht also nur dort, wo das Standardsystem Schnittstellen zu ihrer Datenbank bedient.

Updates der Formulardefinitionen können aus dem Testsystem leicht durch einen Update der XML-Daten im Produktionssystem durchgeführt werden. Das geht in der Regel leichter als eine relationale Datenmodelländerung.

Der Weg des Formulars aus der PL/SQL-Schicht bis zur HTML-Darstellung auf dem Browser-Bildschirm ist schlank und nicht mit Performanceverlusten behaftet, denn Oracle’s modPLSQL-Technik ruft ohne Umweg direkt PL/SQL-Prozeduren auf, die in unserem Fall mit dem Formular-API den HTML-Code der Formularfelder generiert.

Kontaktadresse:

Enterprise Web AG
Tonhallenstraße 19
D-47051 Duisburg

Telefon: +49(0)203-295 25 90
Fax: +49(0)203-295 25 99
E-Mail info@webag.com
Internet: www.webag.com