Kobra, übernehmen Sie!

Endbenutzer an den Schalthebeln von Apex

Ingrid Hayek
Universität Innsbruck
Innsbruck, Österreich

Schlüsselworte
Apex-Applikationen modifizieren, Labels, Hilfetexte ändern, Conditional Display, dynamische Defaulttexte für Endbenutzer

Einleitung

> *„Guten Morgen, Mister/Mrs EndbenutzerIn...*
> *Sollten Sie oder jemand aus Ihrem Betrieb an der Applikation kleine Änderungen vornehmen wollen, wird der Applikationsentwickler jegliche Kenntnis dieser Operation abstreiten. Die Applikation wird sich dabei aber keineswegs selbst vernichten.*
> *Viel Glück, Mister/Mrs EndbenutzerIn. Kobra, übernehmen Sie!“*
> (Mission Impossible - adaptierte Version von IH, 2013)

Die Datenbank ist erstellt, die maßgeschneiderte Applikation steht, der Kunde ist zufrieden.
So weit so gut.
Aber Zeiten ändern sich, Ansprechpersonen wechseln, neue Bedürfnisse entstehen, Anforderungen werden angepasst. Der Kunde kann die Inhalte der Datenbank dynamisch und individuell verändern und ergänzen - wie aber steht es um das relativ starre "Gerüst" einer Apex Applikation?

Der Applikationsentwickler muss sich Jahre nach Fertigstellung bei auftauchenden Änderungswünschen wieder einarbeiten und empfindet diese Arbeit oft als lästig. Der Kunde muss als Bittsteller auftreten, beziehungsweise für Änderungswünsche bezahlen, die er als "Kleinigkeiten" empfindet.
Was spricht dagegen, dem Kunden mehr Macht über die für ihn entwickelte Applikation zu geben, wenn man verhindern kann, dass er dabei Schaden anrichtet?

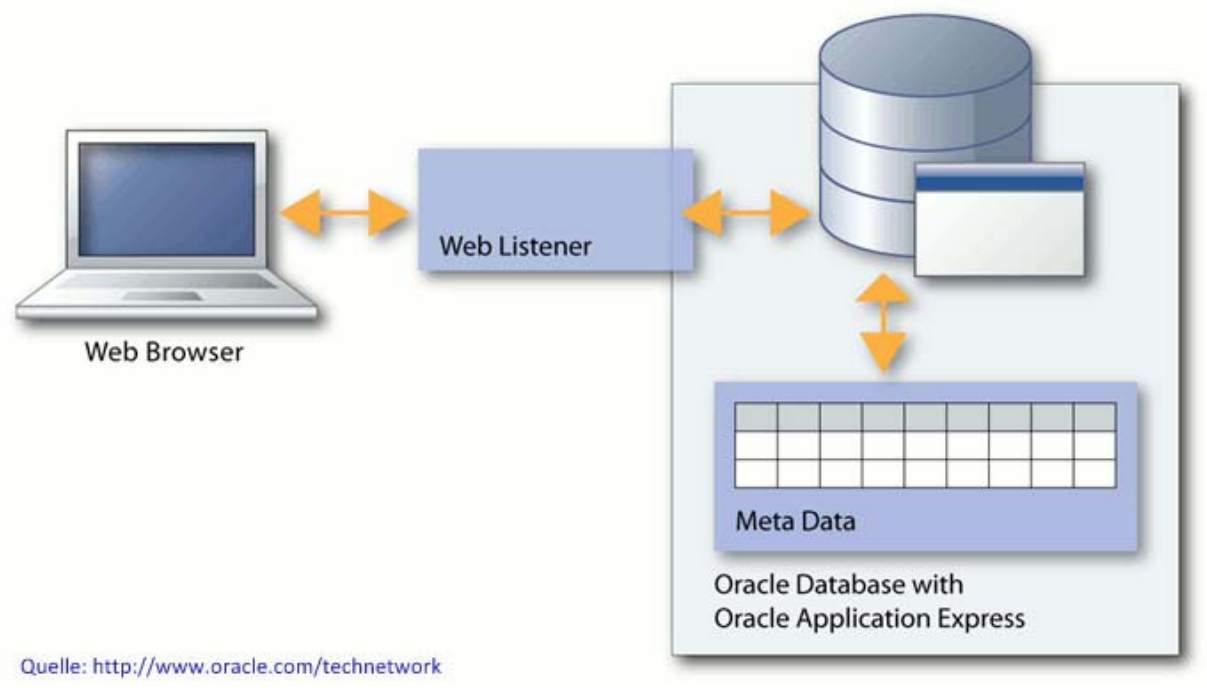
Der Vortrag zeigt Tipps und Tricks, wie man den Kunden Möglichkeiten bieten kann, Apex Webformulare selbst individuell anzupassen:

- Item Labels selbst adaptieren.
- Nach Lust und Laune Default-Texte festlegen.
- Persönliche Hilfetexte erstellen.
- Bestimmte Items je nach Wunsch oder Berechtigung des eingeloggten Benutzers ein- oder ausblenden...
- ... und als Entwickler gleichzeitig lästige Nachbesserungsarbeiten los zu sein!

Kein Problem, wenn man nur weiß, wie.

Application Express Architektur
Apex ist ein "Application Framework", das heißt ein Programm zur Applikationswentwicklung, das selbst mit Apex erstellt wurde.
Ein wichtiges Merkmal von APEX ist dabei, dass kein ausführbarer Code erzeugt wird, sondern die Anfragen im Browser direkt in PL/SQL-Aufrufe umgesetzt werden. Dabei werden die Webapplikationen, die in Datenbanktabellen gespeichert sind, in Echtzeit aus Metadaten "gerendert".

In diesem Metadaten Repository werden sämtliche Definitionen von Applikationen gespeichert.



Das Repository "lebt" komplett innerhalb unserer Oracle-Datenbank und besteht lediglich aus einer Unmenge von Daten, die in Tabellen gespeichert sind, sowie einem riesigen PL/SQL Code. Hinter Apex verbergen sich zirka 425 Tabellen und 230 Packages mit insgesamt mehr als 425000 Zeilen PL/SQL-Code.

Was bedeutet das für den Entwickler?
Zunächst einmal: oh, kompliziert! (siehe oben).
Dann aber: Wenn Apex die Daten selbst in der Datenbank speichert, muss es eine Möglichkeit geben, darauf zuzugreifen. Nun gibt Oracle die Tabellen, in denen die Daten der Apex-Entwicklungsumgebung gespeichert werden, natürlich nicht frei - zu groß wäre die Gefahr, dass bei Änderungen das System bricht.
Sehr wohl aber haben Entwickler Lese-Zugriff auf die Daten, und zwar in Form von Views.
Von diesen Views nun können wir uns genau diejenigen näher ansehen, die wir für die Adaptierungen unserer Applikation brauchen.

Dynamische Labels
Betrachten wir zunächst einmal den View[1] apex_application_page_items:

Hier finden wir sämtliche Applikationen mit allen Definitionen zu den Items:
```
SELECT * FROM apex_application_page_items
```

So finden wir sämtliche Definitionen zu den Items einer bestimmten Applikation:
```
SELECT * FROM apex_application_page_items
 WHERE application_name = 'KOBRA'
```

[1] Es ist gängig, den englischen Begriff View für eine Darstellungskomponente zu verwenden. Allerdings ist es nicht völlig klar, ob der Artikel nun als "der View" oder "die View" zu wählen ist. Ich verwende die Version "der View" (http://www.duden.de/suchen/dudenonline/View)

Und jetzt die relevanten Definitionen zu den Items, die wir für dynamische Formulare brauchen:

```
SELECT application_name, page_id, item_name, display_as, label,
       condition_type, condition_expression1, item_default
  FROM apex_application_page_items
 WHERE application_name = 'KOBRA'
```

Item Name und zugehöriges Label werden also im Repository gespeichert.

Wie kann nun Apex dazu gebracht werden, Labels gemäß unseren Wünschen dynamisch zu produzieren?

1. Applikation erstellen oder vorhandene Applikation auswählen, zum Beispiel `'KOBRA'`.

2. alle dynamischen Labels mit "LABEL" beschriften.

3. eventuell schon Hilfetexte eingeben

4. Tabelle mit den essentiellen Informationen aus dem Repository erstellen, die der Endbenutzer editieren kann

   ```
   CREATE TABLE apexadmin AS
   SELECT application_id, page_id, item_name,
          label AS label_orig, label,
          item_default,
           condition_type, condition_expression1, item_help_text
     FROM apex_application_page_items
    WHERE application_name = 'KOBRA'
      AND label = '"LABEL"'
      AND display_as != 'Hidden'
      AND condition_type != 'Never'
      AND authorization_scheme != 'Developer'
   ```

5. Ein Shortcut, um Apex mitzuteilen, wo es die Information für die Labels holen soll:
   RETURN apex.get_label('#CURRENT_ITEM_NAME#');

6. Function get_label:

   ```
   FUNCTION get_label(p_item_name IN VARCHAR2) RETURN VARCHAR2 IS
      lv_text VARCHAR2(100);
      BEGIN
         SELECT NVL(label, p_item_name)
           INTO lv_text
           FROM apexadmin
          WHERE item_name = p_item_name;
         RETURN lv_text;
      END get_label;
   ```

7. Daten synchronisieren:
   bei Änderungen muss der Applikationsentwickler die Daten synchronisieren. Wird zum Beispiel einem Item ein statisches Label gegeben, muss dieses Item aus der Liste, die dem Endbenutzer zur Verfügung steht, verschwinden.

Dynamische Hilfetexte
Hilfetexte für die (technische) Bedienung einer Apex-Applikation braucht es nicht: Eine benutzerfreundliche Applikation sollte selbsterklärend sein.
Hilfetexte zum *Inhalt* sollten generell nicht vom Entwickler erstellt werden.
Erstens bezieht sich die Hilfe auf den Inhalt der Datenbank, das heißt auf das jeweilige Fachgebiet, und mit dem kennt sich der Endbenutzer mit Sicherheit besser aus als der Entwickler.
Zweitens können sich solche Hilfetexte im Lauf der Zeit ändern, erweitert oder auch hinfällig werden.
In diesem Fall soll der Endbenutzer einfach die Hilfetexte ändern können.
Für die Verwaltung der Hilfetexte verwenden wir dieselbe Tabelle wie für die Labels.

Dynamische Default Werte
Default Value Type: PL/SQL Function
Beispiel:

- entsprechende Function aufrufen:
  ```
  kobra_util.get_user_default('P20_TITLE');
  ```

Dynamische Bedingungen
Condition Type: Exists
Expression1 Beispiele:

- ```
  SELECT 1
    FROM apexadmin
   WHERE item_name = : P1_TITLE AND condition = 'Y'
  ```

- ```
  SELECT 1
    FROM apexadmin
   WHERE item_name = :P1_TITLE
     AND condition = 'Y' AND :APP_USER = 'ADMINISTRATOR'
  ```

Dynamische Informationen bearbeiten
Welche Möglichkeiten biete ich nun dem Endbenutzer, um die Schalthebel von Apex mitzubedienen?

Expertenmodus:
Eine Seite mit Tabular Form, in der alle Items - ähnlich wie in Apex selbst - bearbeitet werden können

Einfacher Modus:

1. Form mit den Feldern , die der Endbenutzer editieren darf, erstellen

2. Mit einem entsprechenden Plugin erscheint dieses Formular in einem Modal Window:
   - Plugin (modal_page.zip) herunterladen:
   - http://skillbuilders.com/oracle-apex/Application-Express-APEX-Consulting-Training.cfm?tab=free-plugin-downloads
   - modal_page.zip entpacken: dynamic_action_plugin_com_skillbuilders_modal_page.sql, in Apex importieren und Plugin installieren

3. Dynamic Action für editierbares Item konfigurieren:
   jQuery Selector: .uHelpLink**Dyn**
   True Action:
   Identification:
   Action aus Select List Effect auswählen: SkillBuilders Modal Page (1.0.0) [Plug-in]

Settings:
Url Location: Attribute of Triggering Element
Attribute Name: href
Auto-close On Element Selector: div#close-modal
Dialog Height/Width Mode: Auto

4. Frames für Modal Window erlauben:
   Application Properties > Security > Browser Security: Enable Frames

5. Label Templates modifizieren:
   Template *Optional with help /Required with help* - Definition: Before Label:

```
<label for="#CURRENT_ITEM_NAME#" tabindex="999" class="uOptional/uRequired">
      <a onclick="javascript: return false;" class="uHelpLinkDyn"
      href="f?p=&APP_ID.:10002:&APP_SESSION.::::
      P100002_ITEM_NAME:#CURRENT_ITEM_NAME#" tabindex="999" >
```

6. Conditional Display im Modal Window
   Endbenutzer mit dem Status "Administrator" können alle editierbaren Informationen sehen und bearbeiten.
   "Normale User" sehen nur bestimmte Informationen, zum Beispiel den Hilfetext, und können diese Informationen nicht editieren (Read Only Condition)

So, und jetzt lehnen wir uns zurück…

Client:..*And Mr/Mrs Developer, the next time you go on holiday, please be good enough to let us know where you're going.*
Developer: *If I let you know where I'm going, I won't be on holiday.*
(Mission Impossible - adaptierte Version von IH, 2013)

Kontaktadresse:
Dr. Ingrid Hayek
daten & worte
Höhenstraße 110
A-6020 Innsbruck

Telefon: +43 (0) 650-2923290
E-Mail work@ingridhayek.at
Internet: ingridhayek.at