Mehrsprachigkeit in Apex-Anwendungen

Stefan Röß, BASF-Gruppe

Um den zeitlichen und finanziellen Mehraufwand von Übersetzungen gering zu halten, einigt man sich in den Anforderungsspezifikationen zumeist darauf, die Anwendung in Englisch zu betreiben. Doch nicht nur in großen internationalen Unternehmen, auch in kleinen und mittelständischen Firmen sprechen Mitarbeiter oft unterschiedliche Sprachen. Auch müssen Anwendungen in bestimmten Ländern mehrsprachig angeboten werden, wenn mehr als eine offizielle Landessprache vorhanden ist, wie beispielsweise in der Schweiz, Belgien oder Kanada. BASF bietet eine ausgereifte Methodik an, um Mehrsprachigkeit für Apex-Anwendungen umzusetzen.

Die hier vorgestellte Lösung zeigt eine Best Practice, die erst seit der Apex-Version 4.2.3 möglich ist. Mehrsprachigkeit ist in Apex seither einfach zu haben und stellt deshalb keine nennenswerte Hürde mehr dar. Die BASF-Apex-Standard-Anwendung („BAStA“) lagert das Übersetzen als eine Funktionalität in die Anwendung aus, man übergibt also die Verantwortung an die Personen, die die Fremdsprachen beherrschen. Dabei müssen diese noch nicht einmal fachliche Anwender der Applikation sein. Somit sind die Zeiten schwerverständlicher XLIFF-Dateien, die einen speziellen Editor benötigen, endgültig vorbei. Mithilfe des Apex-API „Apex_LANG“ ist es gelungen, eine einfache, rollenbasierte Methodik zu entwickeln, die als Muster für jede Apex-Anwendung zur Mehrsprachigkeit dienen kann.

Abbildung 1: Beispiel für ein Nutzer-Interface

Create	Anlegen der Übersetzungssprache, wird nur einmal durchgeführt
Seed	Anlage der Texte in einer Apex-Tabelle
Translate	Übersetzen der Texte
Publish	Veröffentlichen der übersetzten Texte

Tabelle 1

Was muss übersetzt werden?

Jede neue Apex-Anwendung innerhalb der BASF wird mit dem BAStA-Framework erstellt. Dabei ist es immer möglich, eine oder mehrere Sprachen nachträglich zu implementieren. Übersetzt werden alle Anwendungsseiten des Nutzer-Interface und die Status-Nachrichten wie Bestätigungen oder Fehlermeldungen *(siehe Abbildung 1)*.

Tabellendaten, die die Inhalte einer Anwendung betreffen, werden hierbei nicht berücksichtigt. Sollen auch diese Daten übersetzt werden müssen, kann dies durch ein fachliches Datenmodell in Form einer Übersetzungstabelle geschehen. Dies ist unabhängig von Apex und von der jeweiligen Frontend-Technologie.

Übersetzungsprozess

Das Nutzer-Interface wird in BAStA mit dem integrierten Mehrsprachigkeitsmodul von Apex übersetzt. Dabei wird für jede Sprache eine Art Schatten-Applikation erzeugt.

```
apex_lang.create_language_mapping   (p_applicacion_id                => p_app_id
                                    ,p_language                    => p_lang
                                    ,p_translation_application_id  => p crans app id);
```

Listing 1

Die folgenden vier Schritte müssen für jede Sprache durchgeführt werden und sind in der Anwendung für Endanwender mit der Rolle „Translator“ und „Developer“ verfügbar *(siehe Tabelle 1)*.

Create-Prozess

Der Prozess „Create“ ordnet die Originalsprache (Primary Application Language) der übersetzten Sprache zu. Dies geschieht über einen Eintrag in der Tabelle „wwv_flow_language_map“ des Apex-Schemas. Ausgeführt wird der Eintrag in die Tabelle über die API-Prozedur „create_language_mapping“ *(siehe Listing 1)* aus dem Parsing-Schema.

Abbildung 2 zeigt die Nutzer-Interface-Umsetzung zu *Listing 1*. Die Application ID „10101“ wird hier nicht angezeigt, da sie für die deutsche Sprache reserviert wurde. Wichtig bei der Parameterübergabe aus *Listing 1* ist, dass man „p_language“ einen IANA-Code der Sprache zuordnet. Aktuell sind 133 IANA-Codes unterstützt *(siehe „https://docs.oracle.com/cd/B28359_01/appdev.111/b32258/global.htm“)*. Der Parameter „p_translation_application_id“ kann eine frei wählbare ID sein. Es hat sich jedoch erwiesen, dafür die APPLICATION_ID mit einer angefügten 01 bis 09 zu verwenden.

So wurde in *Abbildung 3* der APPLICATION_ID 101 und dem IANA-Code der deutschen Sprache „de“ die „01“ zugeordnet; somit ergibt sich die ID 10101.

Benötigt man mehr als zehn unterschiedliche Sprachen, werden die IDs 11 bis 19, 21 bis 29, 31 bis 39 etc. angefügt. Die ID darf nicht mit der Zahl „NULL (0)“ enden, da dies zu einem Fehler beim Ausführen von *Listing 1* führt. *Listing 2* zeigt die wichtigsten Parameter nach dem Ausführen von *Listing 1* an.

Abbildung 4 zeigt den Eintrag zu „Application Language Derived From“ = „Item Preference …“. Um diesen Eintrag verwenden zu können, muss eine Applikationsvariable „FSP_LANGUAGE_PREFERENCE“ erstellt werden. Ein Applikationsprozess, der beim Login der Anwendung ausgeführt wird, beziehungsweise ein Page-Prozess, der im Nachgang vom Endanwender aktiv angestoßen werden kann, setzt diese Variable auf den gewünschten IANA-Code und damit auf die Zielsprache des Endanwenders. Ist die Übersetzung eines Textes nicht vorhanden, wird zunächst in der darüber liegenden Sprache der Eintrag gesucht, beispielsweise von Frankokanadisch (fr-ca) zu Französisch (fr). Ist diese Übersetzung auch nicht vorhanden, wird die Originalsprache der Anwendung („Application Primary Language“) verwendet.

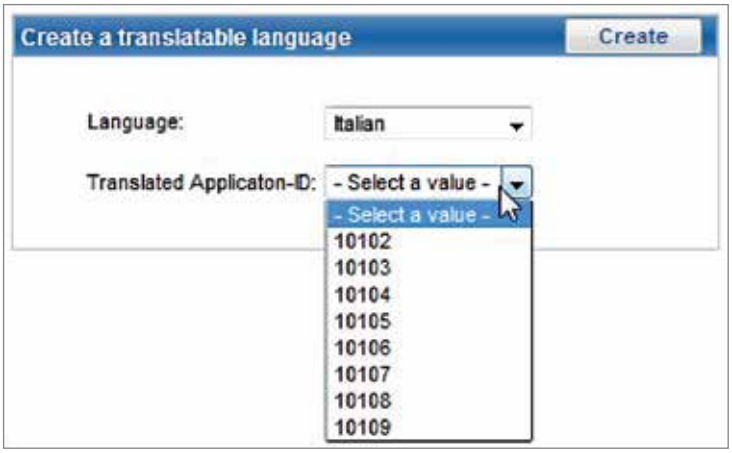

Abbildung 2: Anlegen einer Translation Application ID

```
SELECT a.workspace
      ,a.application_id
      ,m.translated_application_id
      ,a.application_primary_language
      ,m.translated_app_language
      ,a.language_derived_from
  FROM apex_applications a, apex_application_trans_map m
 WHERE 1 = 1
   AND m.primary_application_id = a.application_id;
```

Listing 2

Seed-Prozess

Im Prozess-Schritt „Seed“ werden Texte der Originalsprache des Nutzer-Interface in der Apex-Tabelle „wwv_flow_translatable_text$“ angelegt. Die folgenden Apex-Schema-Objekte sind davon betroffen. Dabei wird die Tabelle über die Prozedur „wwv_flow_lang.

WORKSPACE	APPLICATION_ID	TRANSLATED_APPLICATION_ID	APPLICATION_PRIMARY_LANGUAGE	TRANSLATED_APP_LANGUAGE	LANGUAGE_DERIVED_FROM
BASTA	101	10101	en-us	de	ITEM_PREFERENCE
BASTA	101	10102	en-us	it	ITEM_PREFERENCE

Abbildung 3: Die Ergebnisse aus Listing 2 – man sieht den IANA Code „de“ und „it“

seed_translations“ und mit Bezug auf die Meta-Tabelle „wwv_flow_translatable_cols$“ mit allen übersetzbaren Werten befüllt. Man spricht hier von Apex-internen Vorgängen, die über die API-Prozedur „seed_translations“ aus dem „Parsing-Schema“ heraus ausgeführt werden *(siehe Listing 3)*.

Translation-Prozess

Jetzt werden über das Parsing-Schema der API-Prozedur „update_translated_string“ die Werte der Spalte „translate_to_text“ aus der Tabelle „wwv_flow_translatable_text$“ des Apex-Schemas geändert *(siehe Listing 4)*.

Texte können auf zwei Arten übersetzt werden: entweder über eine Einzeldatensatz-Verarbeitung, etwa für nachträgliche, punktuelle Anpassungen, oder über einen Excel-Download zur Massenverarbeitung und anschließenden Excel-Upload, etwa für das initiale, erstmalige Übersetzen *(siehe Listing 5 sowie Abbildungen 5)*.

Publish-Prozess

Die Übersetzungen werden durch „publish“ live geschaltet. Dies geschieht zunächst über einen internen Kopiervorgang, das Erzeugen der sogenannten „Schatten-Applikation“. Dabei sind unter anderem die folgenden Apex-Schema-Objekte betroffen: Zunächst wird über die Prozedur „wwv_flow_translation_util_api.flow_copy“ die neue Sprachen-ID im Language-Repository angelegt. Dann werden die übersetzten Texte mithilfe der Prozedur „wwv_flow_translation_util_api.sync_translations“ aus der Tabelle „wwv_flow_translatable_text$“ auf die jeweiligen Tabellen des Language-Repository übertragen. Als Metadaten-Referenz gilt hier ebenfalls „wwv_flow_translatable_cols$“. Dies sind interne Vorgänge im Apex-Schema, die über die API-Prozedur „publish_application“ aus dem Parsing-Schema heraus gestartet werden *(siehe Listing 6)*. *Abbildung 5* zeigt oben in der Leiste den Publish-Button, der das API „apex_lang.publish_application“ ausführt.

Löschen einer Sprache

Im Löschvorgang werden alle notwendigen Daten aus den Tabellen des Sprachen-Repository mit der Prozedur „delete_language_mapping“ gelöscht. *Listing 7* zeigt den Aufruf aus dem Parsing-Schema.

Abbildung 4: Globalisierungsparameter der Anwendung 101

```
apex_lang.seed_translations (p_applicacion_id     => p_app_id
                            ,p_language           => lv_trans_lang);
```

Listing 3: Seeding der Sprache. Dem Parameter „p_language“ wird der IANA-Code der zu übersetzenden Sprache übergeben („lv_trans_lang“)

```
apex_lang.update_translated_string (p_id          => p_id
                                   ,p_language    => lv_trans_lang
                                   ,p_string      => p_to_string);
```

Listing 4

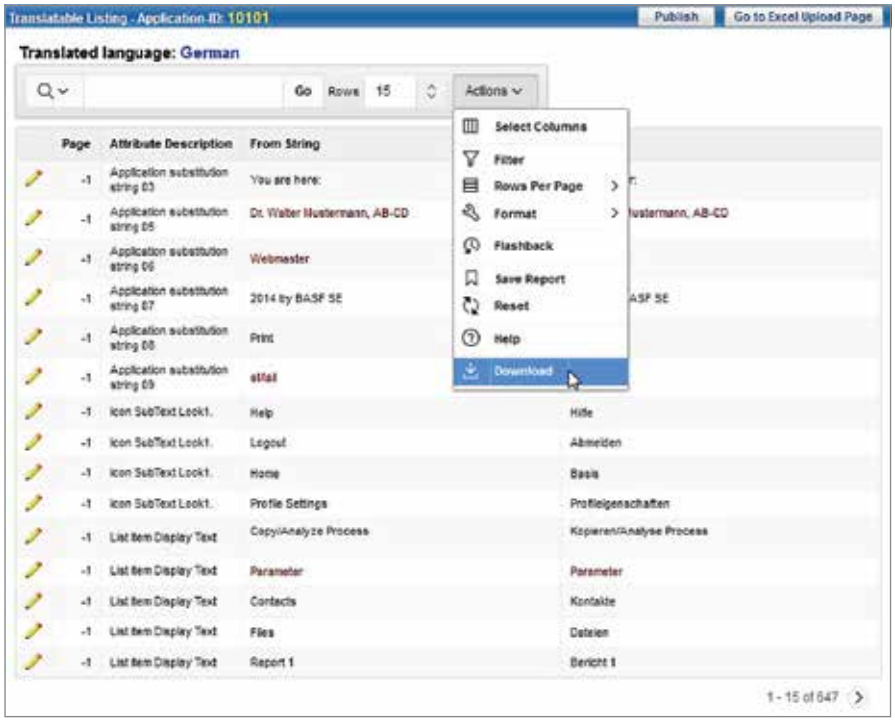

Abbildung 5: Report auf die Apex-View „apex_application_trans_repos“ mit der Möglichkeit zum Download für Massenverarbeitung oder über das Stift-Icon zur Einzeldatensatz-Verarbeitung

Anlegen von Translatable Messages

Mit den Translatable Messages sind Nachrichten wie Bestätigungen oder Fehlermeldungen gemeint. Sie können aber auch für alle anderen Arten von Text Verwendung finden. Nachrichten werden in die Tabelle „wwv_flow_messages$" des Apex-Schemas eingefügt und können über die View „APEX_APPLICATION_TRANSLATIONS" des Parsing-Schemas ausgewählt werden. Mit dem Aufruf der Prozedur „create_message" werden Nachrichten befüllt *(siehe Listing 8)*.

Aktualisieren von Nachrichten

Um eine bestimmte Nachricht zu verändern, ruft man die Prozedur „update_message" aus dem Parsing-Schema auf *(siehe Listing 9)*.

Löschen von Nachrichten

Interessanterweise werden Nachrichten mit der gleichen Prozedur gelöscht wie hinzugefügt. Der entscheidende Unterschied ist der Modus („Global Mode"). Man setzt diesen Modus auf „REMOVE" und übergibt der „create_message"-Prozedur die „translated_entry_id (lv_trans_id)", das ist der eindeutige Schlüssel für den Datensatz *(siehe Listing 10)*.

Darstellung der Nachricht

Eine erstellte Nachricht wird im PL/SQL-Code über die Funktion „apex_lang.message" aufgerufen und in der Applikation dargestellt. Dabei können bis zu zehn verschiedene Parameter dem Identifier mitgegeben werden. In *Listing 11* wurden dem Message-Identifier „CREATE_LANG_MAP_SUCCESS" zwei Parameter übergeben. Diese Parameter flexibilisieren die zu erzeugende Nachricht *(siehe Abbildung 6)*.

Ist der Application Builder in weiteren Sprachen installiert, was über die SQL-Skripte „load_language.sql" wie „load_de.sql" oder „load_it.sql" möglich ist, sind die übersetzten Texte für den Interactive Report und die Paginierung der Reports enthalten. Ist dies nicht der Fall, sind die „APEXIR_*"-Texte wie „APEXIR_SELECT_COLUMNS"," APEXIR__FILTER" und „APEXIR__ROWS_PER_PAGE" sowie die Pagination-Texte wie „PAGINATION.PREVIOUS_SET" in den Messages für die jeweilige Sprache zu übersetzen *(siehe „https://docs.oracle.com/cd/B28359_01/appdev.111/b32258/global.htm" und Abbildung 7)*.

```
FOR c
   IN (SELECT seq_id
         ,replace (c002, '''') c002
         ,c003
         ,c004
         ,c005
         ,c006
      FROM apex_collections
      WHERE collection_name = p_coll_name AND cOOl = p_excel_name AND
      seq_id != 1)
LOOP
   Apex_lang.update_translated_string   (p_id          => c.c002
                                        ,p_language    => lv_trans_lang
                                        ,p string      => c.c006);
END LOOP;
```

Listing 5: Massenverarbeitung erfolgt über eine „Apex_COLLECTION"

```
apex_lang.publish_appiication (p_application_id=> p_app_id
                              ,p_language       => lv_trans_lang);
```

Listing 6: Live-Schalten einer bestimmten Mapping-Language

```
apex_lang.delete_language_mapping   (p_application_id  => p_app_id
                                    ,p_language        => lv_trans_lang);
```

Listing 7: Löschen einer bestimmten Mapping Language

```
wwv_flow_api.create_message    (p_flow_id           => p_app_id
                               ,p_name              => upper (p_message_name)
                               ,p_message_language=> lower (p_target_lang)
                               ,p_message_text      => p_message_text);
```

Listing 8: Call von „create_message" vom Parsing-Schema. Der Parameter „p_message_name" legt den Message-Identifier für eine bestimmte Sprache „p_target_lang" fest. In „p_message_text" steht der übersetzte Text

Abbildung 6: Der „CREATE_LANG_MAP_SUCCESS"-Message-Identifier

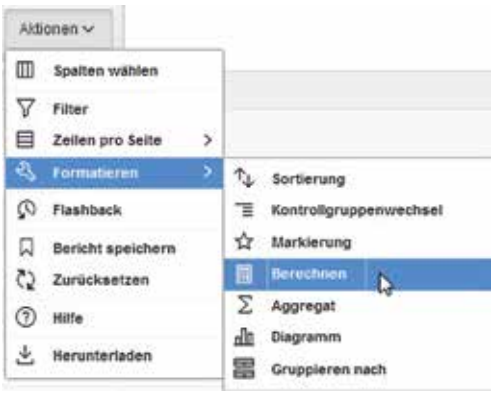

Abbildung 7: IR-Report-Action-Button und dessen Einträge

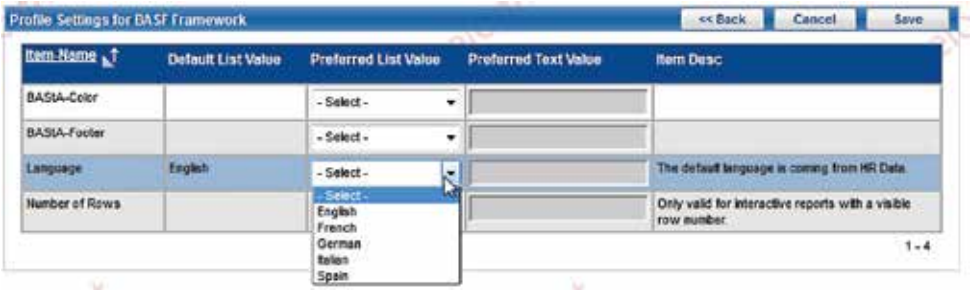

Abbildung 8: Umschalten in eine andere Sprache

```
apex_lang.update_message (p_id              => lv_trans_id
                         ,p message text    => p_upd_message_text);
```

Listing 9

```
wwv_flow_api.g_mode := 'REMOVE';
wwv_flow_api.create_message (p_id => lv_trans_id);
```

Listing 10

```
apex_lang.message    ('CREATE_LANG_MAP_SUCCESS'
                     ,lv_lang
                     ,p-trans_app_id);
```

Listing 11

Auswahl der gewünschten Nutzer-Interface-Sprache

Für jeden neuen Anwender wird die jeweilige Sprache in einer Userprofile-Tabelle festgehalten. Meldet sich ein Anwender an, erfolgt ein „Post Authentication Process", der das Application-Item „FSP_LANGUAGE_PREFERENCE" setzt. In den Globalisierungsparametern wurde deshalb unter „Shared Components" die „Application Derived Language" auf „Item Preference ..." eingestellt *(siehe Abbildung 6)*. Nachdem alle Sprach-Komponenten übersetzt sind, kann jeder sein eigenes Sprachenprofil auch verändern. Dazu wird über einen Page-Prozess „FSP_LANGUAGE_PREFERENCE" verändert *(siehe Abbildung 8)*.

Auf die API-Methode „apex_lang.lang" („Klavierspieler-Methode") hat BASF ganz verzichtet. Damit könnten Werte aus dynamischen Lists of Values (LOV) und Werte aus anderen Tabellen übersetzt werden. Da dies nicht zu den typischen Eigenschaften eines Nutzer-Interface zählt, werden Übersetzungen in einem solchen Fall von einem fachlichen Datenmodell berücksichtigt.

Applikationserweiterungen

Im Laufe der Zeit ändern sich Anwendungen aufgrund neuer Anforderungen. In diesem Fall wird die Applikation ganz normal über den Application Builder weiterentwickelt, dies geschieht immer in der „Application Primary Language".

Zu jeder Zeit können die Prozesse „Seed", „Translate" und „Publish" durchgeführt werden, sodass neue Items, Regionen und Pages in der zu übersetzenden Sprache dargestellt beziehungsweise gelöscht werden, wenn sie obsolet sind. Das interne Mehrsprachigkeitsmodul von Apex übernimmt dafür die nötigen Schritte.

Export und Import übersetzter Applikationen

Die übersetzten Nachrichten („Translated Messages") werden bei einem Export immer einbezogen und stehen bei einem Import der Anwendung sofort zur Verfügung.

Dabei spielt die Einstellung der Export-Einstellungen keine Rolle. Das Export-Translation-Flag ist lediglich für das Sprachen-Repository gültig, das mit dem internen Mehrsprachigkeitsmodul („Create", „Seed", „Translate", „Publish") angelegt wurde. Damit lässt sich steuern, ob alle Übersetzungen exportiert werden sollen. Hat man beim Export das Flag auf „Yes" gesetzt, erhält man beim anschließenden Import in eine andere Datenbank eine neue Schatten-Applikations-ID („Translated Application ID"). Dabei wird die nächste freie Nummer gezogen. Das ist nicht weiter tragisch, nur sollte man es wissen. Aber auch hier gibt es Möglichkeiten, die gleiche ID zu behalten.

Fazit

Das Mehrsprachigkeitsmodul von Apex ist mit seinen bestehenden Bibliotheken einfach in eine Applikation einzubinden, um daraus ein eigenes Framework aufzubauen. Der Entwickler kann sich daraufhin voll und ganz um die Umsetzung der fachlichen Anforderungen kümmern. Übersetzungen werden von den Kunden beziehungsweise der Fachseite selbst vorgenommen. Mit den bestehenden API-Methoden kann man effizient Mehrsprachigkeit umsetzen. Das Apex-Framework der BASF ist somit mehr als konkurrenzfähig für multilinguale Anwendungen gegenüber anderen Software-Produkten.

Stefan Röß
stefan.roess@basf.com