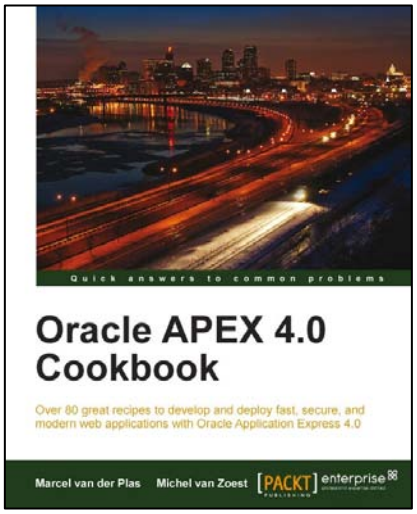
[PACKT] PUBLISHING

Oracle APEX 4.0 Cookbook

Marcel van der Plas
Michel van Zoest



Chapter No. 6
"Creating Multilingual APEX Applications"

In this package, you will find:

A Biography of the authors of the book

A preview chapter from the book, Chapter NO.6 "Creating Multilingual APEX Applications"

A synopsis of the book's content

Information on where to buy this book

About the Authors

Marcel van der Plas has been an Oracle Consultant for over 15 years. And from the beginning, he learned to work with Oracle Forms, Oracle Reports, and Oracle Designer. Marcel has worked on many projects with these tools. Later on, he became interested in APEX and did some projects with APEX.

Marcel currently works for Ciber. Other companies he worked for are Atos Origin and Whitehorses. For Whitehorses, he wrote some articles ("Whitebooks") about Oracle.

I would like to thank Michel van Zoest, my co-author for helping and working together. I also want to thank the reviewers Maarten van Luijtelaar, Dimitri Gielis, and Surachart Opun. Their comments were so valuable and helpful. I would like to thank Douwe Pieter van den Bos for introducing us to Packt and I would like to thank Packt for giving me this opportunity to write this book. I would like to thank my employer, Ciber, for giving me the freedom to write this book.

Last but not least, I would like to thank my wife Yvonne and my children Vera, Laura, and Joey for inspiring and supporting me. At the same time, I would like to apologize to them for not having time to play on the weekends during the months that I wrote this book.

Michel van Zoest is a consultant with more than 10 years of experience in building (web) applications using Oracle technology such as Oracle (web) Forms, Oracle Designer, MOD_PLSQL, ADF, SOA Suite and of course, APEX.

He is one of the first Oracle Application Express Developer Certified Experts in the world.

He has used his APEX knowledge in projects for companies ranging in size from a single employee to large multinationals. His experience in these projects has been used in the realization of this book.

Michel currently works at Whitehorses in the Netherlands and runs his own blog at `http://www.aboutapex.com`. Next to that, he blogs at the company website on `http://blog.whitehorses.nl` and he regularly writes Whitebook articles (in Dutch) for Whitehorses.

First of all, I would like to thank my co-author Marcel van der Plas. Thanks to the easy way that we could work together, the writing of this book has gone as smooth as possible.

I would like to thank the people at Packt Publishing for offering me the chance to write this book. It has been a long process with a lot of hard work, but I'm very happy with the result. I also would like to thank Douwe Pieter van den Bos for introducing me and Marcel to Packt and his invaluable help in the early stages of the book.

Furthermore, I would like to thank Maarten van Luijtelaar, Dimitri Gielis, and Surachart Opun for their hard work in reviewing our drafts. This book has become so much better thanks to you guys.

I also would like to thank my employer Whitehorses for the support I have
been given.

And last but not least, I would like to thank my family for their love and support. Without the help of my wife Jamila and the "dikke kroelen" from my daughters Naomi and Aniek, this result would not have been possible.

Oracle APEX 4.0 Cookbook

Oracle Application Express 4.0 is a rapid web application development tool that works with the Oracle database. Using features like Plug-ins and Dynamic Actions, APEX helps you build applications with the latest techniques in AJAX and JavaScript.

The Oracle Application Express 4.0 Cookbook shows you recipes to develop and deploy reliable, modern web applications using only a web browser and limited programming experience.

With recipes covering many different topics, it will show you how to use the many features of APEX 4.0.

You will learn how to create simple form and report pages and how to enhance the look of your applications by using stylesheets. You will see how you can integrate things such as Tag Clouds, Google Maps, web services, and much more in your applications. Using Plug-ins, Dynamic Actions, BI Publisher, Translations, and Websheets, you will be able to enhance your applications to a new level in APEX.

This book will show you how to be agile in the development of your web applications by using Team Development, debugging, and third-party tools.

After reading this book, you will be able to create feature-rich web applications in Application Express 4.0 with ease and confidence.

What This Book Covers

Chapter 1, Creating a basic APEX application, describes the basic steps to create an APEX application. We will learn to make an intranet application where employees can get information.

Chapter 2, Themes and Templates, presents some recipes which will make your application look better using themes and templates by creating your own theme, including images in it and so on.

Chapter 3, Extending APEX, shows us how to we will extend our application with some nice features like visual effects, a tag cloud, and a Google map.

Chapter 4, Creating Websheet Applications, teaches us how to create a websheet application, create a page in the application, add a navigation page to the websheet, and allow multiple users to access the websheet.

Chapter 5, APEX Plug-ins, describes the four types of plug-ins: Item type, Region type, Dynamic action, and Process type plug-ins.

Chapter 6, Creating Multilingual APEX Applications, shows us how we can fully translate an application using built-in functionality to translate applications, without having to rebuild the application completely and adding something of ourselves to easily switch between languages.

Chapter 7, APEX APIs, shows us how to use APIs as they offer a lot of flexibility and speed in developing web applications.

Chapter 8, Using Webservices, teaches us how to use webservices in APEX.

Chapter 9, Publishing From APEX, shows you how to export reports and get the output in some kind of digital format and how to interact with BI Publisher.

Chapter 10, APEX Environment, contains recipes that will show how to set up and use a development environment, how to use version control and how to deploy Application Express on a web container with the APEX Listener.

Chapter 11, APEX Administration, shows you how to create a workspace, how to create users on the workspace and how to manage the workspaces.

Chapter 12, Team Development, we will see how we can take advantage of the features in Team Development in our project. Each recipe will show how a part of Team Development can be put to use in a specific part of the project cycle.

6

Creating Multilingual APEX Applications

In this chapter, we will cover the following topics:

- Creating a translatable application
- Using XLIFF files
- Switching languages
- Translating data in an application

Introduction

The consequence of publishing applications on the web is that anyone from anywhere in the world (and beyond since they have the Internet on the International Space Station) can view your work. When your target audience is in one country, a single language is often enough. But what can you do when you want to serve your website to many international visitors in their native tongue?

Application Express offers built-in functionality to translate applications, without having to rebuild the application completely.

This chapter is about how we can fully translate an application using those built-ins and adding something of ourselves to easily switch between languages.

Creating a translatable application

An application needs to be altered before translations will work. It has to know for instance that it is going to be translated, by setting some properties.

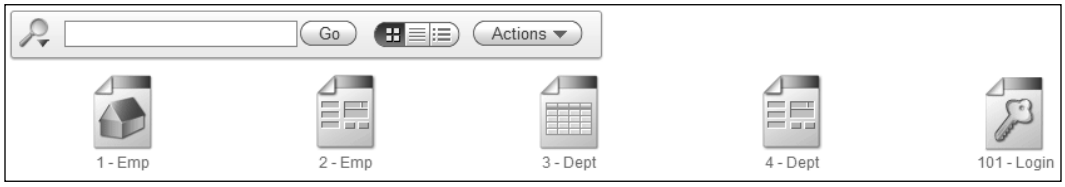
This recipe will show how an application can be prepared for translations.

Getting ready

To start with this recipe we will need a new application. We will use the EMP and DEPT tables to build a straightforward application with minimal effort.

1. In the Application Builder, click **Create** to start making a new application.
2. Select **Database** as the Application Type and click **Next**.
3. Select **From Scratch** and click **Next**.
4. Name the application **HR Multilingual**. Leave the Application ID and other options on their default values and click **Next**.
5. Select the Page Type: **Report and Form**.
6. Enter **EMP** as the **Table Name**.
7. Select the Implementation Classic and click **Add Page**.
8. Repeat these steps to add a Report and Form for the DEPT table.
9. Press **Create** to finalize the application using Theme 1 or click **Next** and follow the wizard to select another theme. When you do that, remember to keep the language on English.

We now have an application with five pages; one Login page and a Report and Form page for both the EMP and DEPT tables.

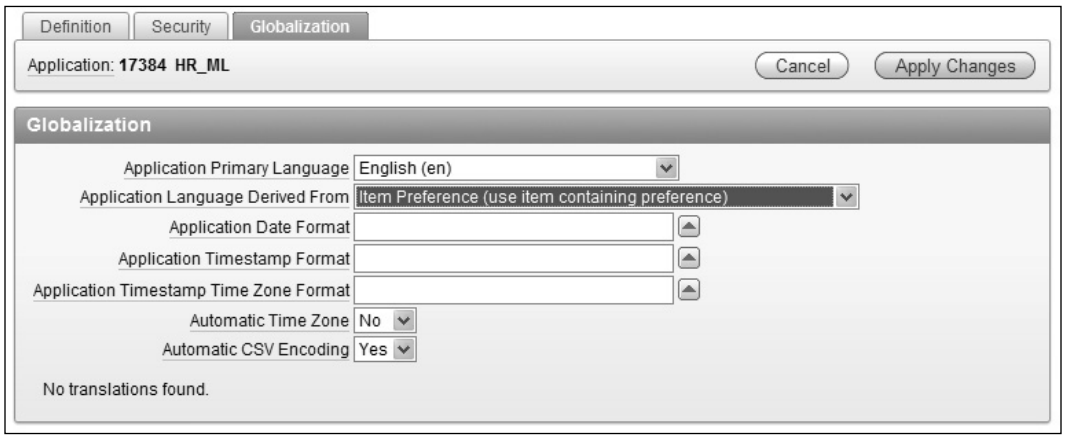


When we run the application, we will see the application in English as can be expected. We are now ready to implement the functionality that is necessary to make this a multilingual application.

How to do it...

Now that we have an application at our disposal, we can start to prepare it for translations. The first step that we have to take is that we have to tell the application how it will know what language to use. This is going to require a special application item.

1. First navigate to the Application overview and click the button that is labeled **Edit Application Properties**.
2. Go to the tab **Globalization**.
3. Set the property **Application Language Derived From** to **Item Preference (use item containing preference)**.



APEX now knows that it can derive the language to show the page in from a special application item. But this item still has to be created. To do this, follow the next steps:

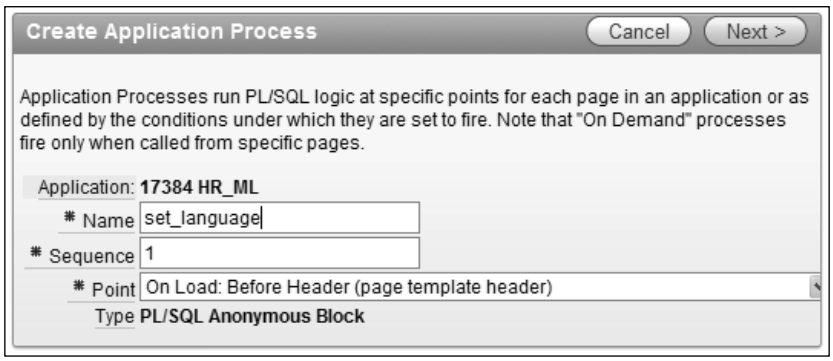
1. Go to **Shared Components**.
2. Under **Logic** click on **Application Items**.
3. Click on the **Create** button.
4. Enter the name **FSP_LANGUAGE_PREFERENCE** for the item and leave all other properties on their default value.
5. Click **Create** to finish.

The item is called FSP_LANGUAGE_PREFERENCE, because APEX recognizes that name as an item reserved for application languages. When a page is rendered, APEX checks the FSP_LANGUAGE_PREFERENCE item to see in what language the page has to be shown.

There is a snag in this process. Because of the way Application Express builds up its page, a change in the FSP_LANGUAGE_PREFERENCE item is not immediately visible. Whenever the language is changed, the page has to be reloaded to show the result.

To make this happen, we will add an Application Process that will handle the reloading of the page:

1. Return to **Shared Components**.
2. Under **Logic** click on **Application Processes**.
3. Click the **Create** button.
4. Enter the name **set_language**.



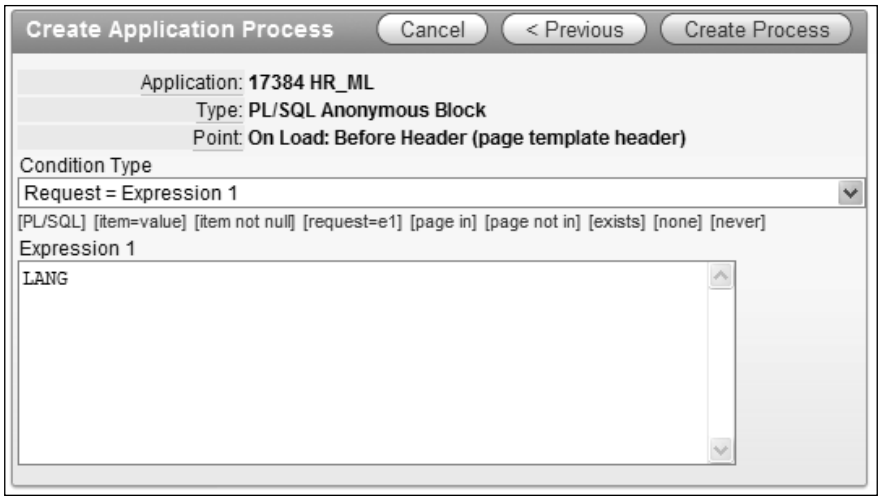
5. Select **On Load: Before Header (page template header)** at the **Point** property to trigger the process as soon as possible on the page.
6. Click **Next**.
7. In the Process Text, enter the following PL/SQL code:

```
begin
owa_util.redirect_url('f?p='||:APP_ID||':'||:APP_PAGE_ID||':'||:
APP_SESSION);
end;
```

8. In the **Error Message** box, enter **Process cannot be executed** and click **Next**.

We choose to start this process when a page is called using a special Request. This will be defined with the following steps:

1. As the **Condition Type**, select **Request = Expression 1**.
2. In the textarea that appears enter **LANG** (all uppercase).



3. Click **Create Process** to finish.

When we now review the list of application processes, we can see the new process has been added to the list.

The application is now ready to be translated. Everything is in place to run it in any language imaginable.

To call the application in another language, change the URL of your application to the following:

```
http://yourdomain:port/pls/apex/f?p=&APP_ID.:&PAGE_ID.:&SESSION_
ID.:LANG:NO::FSP_LANGUAGE_PREFERENCE:nl
```

This example will call the chosen page in the application and show it in the Dutch language instead of in English. To select another language change the property `nl` at the end of the URL to your desired language code.

See also

Now that we have a translatable application, we can start on the translation itself. The application is still only available in English, so we will have to create a translated version of the application in another language.

This will be shown in the recipe called *Using XLIFF files*.

Next to that we can only call the application in other languages by changing the FSP_LANGUAGE_PREFERENCE item in the URL. We should create a more user-friendly way to navigate to different languages.

This topic will be covered in the recipe called *Switching languages*.

Item preference is not the only way of telling an application how to get it's language. A new option in APEX 4.0 is 'Session'. The recipe called *Translating data in an application* will show how to use this. It will also show an example of translating the data in an application.

Using XLIFF files

One of the world's most recognized standards in localization is the XLIFF format. XLIFF is the abbreviation for **XML Localisation Interchange File Format**. As the name explains, it's an XML standard and it has been adopted by Application Express as the format in which different language files can be exported from and imported into APEX applications.

This recipe will show how we can use XLIFF to quickly translate an application from its default language to a new language.

Getting ready

Start with the translatable application that we have built in the first recipe of this chapter. By doing that, we have a good and clean starting point for the first translation.

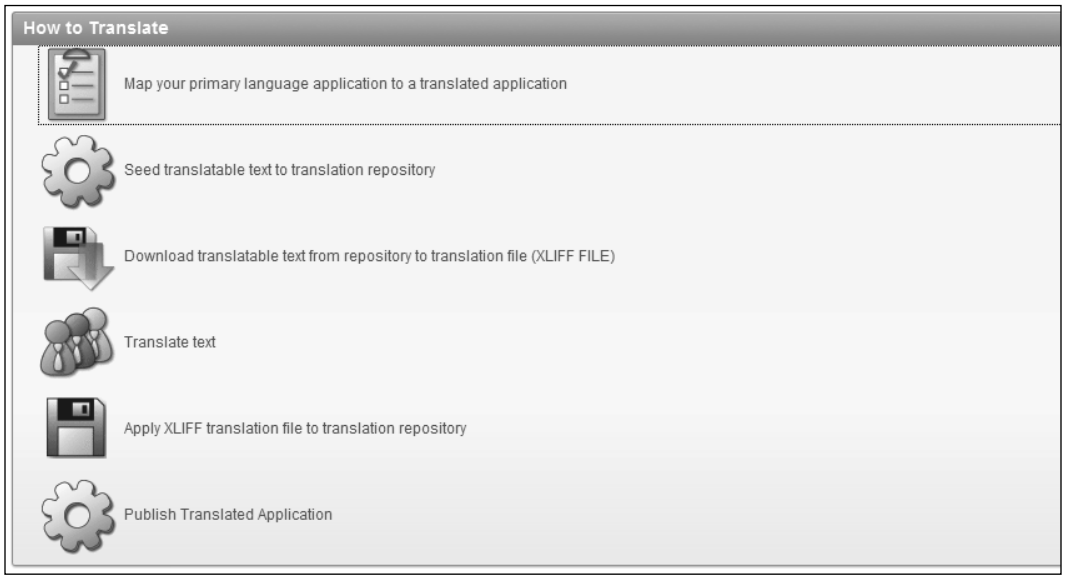
How to do it...

1. Navigate to **Shared Components**.

   At the bottom of the page we can see a section called **Globalization**. This section holds all options that are needed to translate the current application.

2 Click on the link that is labelled **Translate Application**.

This will open up a new page with several options to use when translating an application. In fact, it's a list of all steps necessary to fully translate the application in APEX.

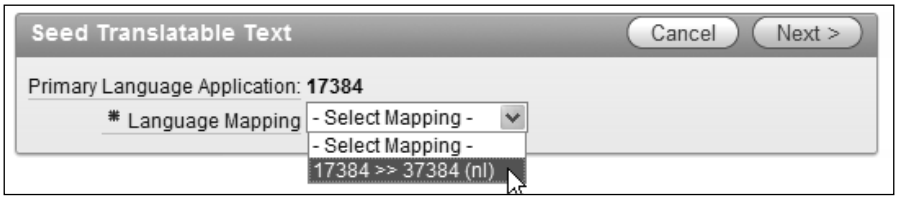


3. Click on the first link **Map your primary language application to a translated application**. This will tell APEX the new language that is going to be used.
4. Click the **Create** button.
5. Enter a new application ID that is not yet used by your APEX environment.
6. Select the language code that you want to translate to. In this example, we will use 'Dutch (Netherlands) (nl)'.

7. Click **Create**.

   The fundament for the new translation is now in place and we can proceed to the next step.

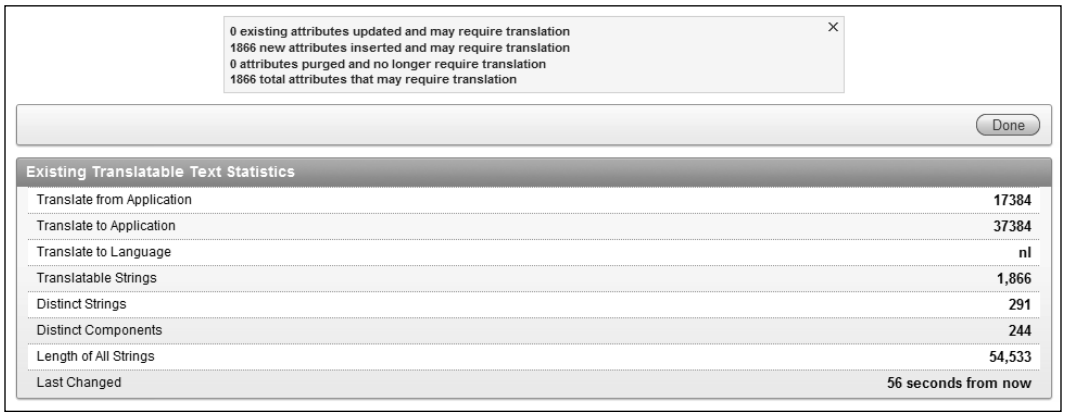
8. Return to the Application Translation Home by clicking on the **Translate** link in the breadcrumb.
9. Click the second step called **Seed Translatable Text** to translation repository.
10. Select the **Language Mapping** that we just created.



11. Click **Next**.
12. Click **Seed Translatable Text**.

This will create entries in the APEX repository for every translatable text in the application. These entries will be used later to generate the XLIFF files.

After the seeding process is completed, APEX shows an overview with summaries of all relevant figures. Click **Done** to close this screen.



1. Return to the Application Translation Home again by clicking on the **Translate** link in the breadcrumb.

2. Click the third step called **Download translatable text from repository to translation file (XLIFF FILE)**.

   The following screen offers two options; download an XLIFF file for the complete application or download a file for a specific page. We will use the second option to get a feeling of what is required to use an XLIFF file.

3. In the bottom section called **Download XLIFF file for Application Page** use the first select list to select the application we are working on.

4. In the second select list select page **101 Login**.

5. Press the button **Export XLIFF**.

6. Use the dialogs to save the file to a location on the hard drive.

   The XLIFF file has now been downloaded and is ready to be edited. This is the fourth step in the Translation process.

7. Use your file browser program to locate the downloaded `.xlf` file and open it for editing in your favourite text editor.

8. Scroll down to the bottom and find the entries with `Username` and `Password`. Change the text between the 'target' tags to the following:

```
<source>Username</source>
<target>Gebruikersnaam</target>

<source>Password</source>
<target>Wachtwoord</target>
```

By changing these two entries, we will at least be able to see the changes. Feel free to change all other translations as well to see the effect on the login page, but this is not necessary for this recipe.

We will now upload the altered file back into APEX.

1. Save the XLIFF file and return to the APEX Application Translation Home.

2. Click on the fifth step called **Apply XLIFF translation file to translation repository**.

3. Click the button labelled **Upload XLIFF**.

4. Enter the title **en_nl_translation** and use the **Browse** button to find the `.xlf` file that we have just edited.
5. Click **Upload XLIFF File**.
6. Click on the title of the XLIFF file we just uploaded.
7. In the next screen, select the translation mapping to apply the XLIFF file to.

8. Click **Apply XLIFF Translation File** to finish connecting the file to the mapping.
9. In the next screen, select the translation mapping again in the select list labelled **Publish Application Translation**. Note that this is equal to Step 6 in the list on the Application Translation Home.
10. Click **Publish Application**.

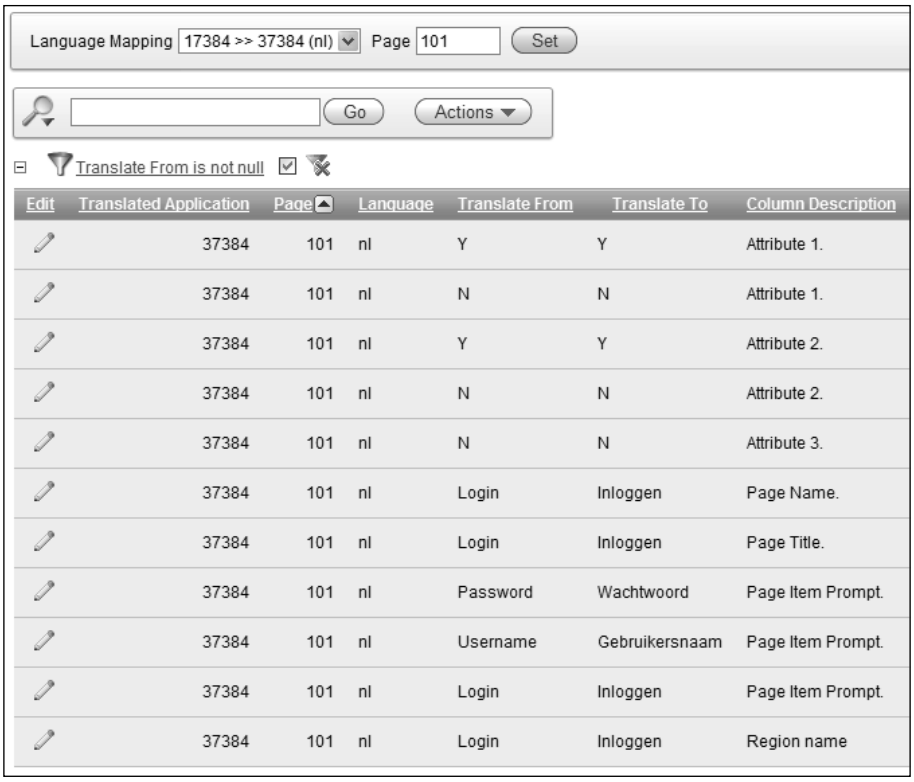
We are done with the translation of all APEX texts in this application.

How it works...

All translatable text is saved in the Application Express Repository. It's possible to directly alter these entries by navigating to a special page.

1. Go to **Shared Components**.
2. Go to **Translate Application**.
3. In the bottom section called **Translation Utilities**, click on the item **Manually Edit Translation Repository**.

From there, we can select the right mapping and if desired a single page. Because this is an Interactive Report, we can use filters and other features to find the desired text to translate.



Edit	Translated Application	Page	Language	Translate From	Translate To	Column Description
	37384	101	nl	Y	Y	Attribute 1.
	37384	101	nl	N	N	Attribute 1.
	37384	101	nl	Y	Y	Attribute 2.
	37384	101	nl	N	N	Attribute 2.
	37384	101	nl	N	N	Attribute 3.
	37384	101	nl	Login	Inloggen	Page Name.
	37384	101	nl	Login	Inloggen	Page Title.
	37384	101	nl	Password	Wachtwoord	Page Item Prompt.
	37384	101	nl	Username	Gebruikersnaam	Page Item Prompt.
	37384	101	nl	Login	Inloggen	Page Item Prompt.
	37384	101	nl	Login	Inloggen	Region name

The quickest way to translate a lot of text is still using the XLIFF file. Because this is an international standard, an XLIFF file can be handled by any translation service.

An XLIFF file is always built the same way; a header and a body. The first part of the header shows some information about the file, the original application, and the translation target application. Because this is commented, it will not be used by a processing application.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  ******************
  ** Source      :  17384
  ** Source Lang:  en
  ** Target      :  37384
  ** Target Lang:  nl
  ** Filename:     f17384_37384_en_nl.xlf
  ** Generated By: MICHEL.VAN.ZOEST@WHITEHORSES.NL
  ** Date:         15-JUN-2010 14:05:34
  ******************
 -->
```

The second part of the header shows the version of the XLIFF standard that is used. The latest specification version is 1.2, but APEX still uses 1.0 by default.

```
<xliff version="1.0">
```

The third part of the header holds some more information about the file, the source language, and the target language. This information is used by APEX when importing the file.

```
<file original="f17384_37384_en_nl.xlf" source-language="en" target-
language="nl" datatype="html">
<header></header>
```

After the header the body starts.

```
<body>
```

The following part is repeated for every text in the APEX repository: an identification for the item that holds the text, a source in the original language, and a target in the new language.

```
<trans-unit id="S-2-27846723888160713-17384">
<source>Logout</source>
<target>Uitloggen</target>
</trans-unit>
```

Finally, the file is closed.

```
</body>
</file>
</xliff>
```

There is one thing that you will always have to keep in mind when using XLIFF files for translating an application. Every time something is changed in the application, you will have to repeat the process of seeding, exporting, translating, and importing. Otherwise, not all changes will be visible in the translated version of the application.

So when the translating itself is done by an external company, it would be wise to send them the XLIFF file after the application in the default language is completely done. This could save a lot of money.

Switching languages

In the past two recipes, we have changed a standard application so that it can be translated and we performed the translation itself. In this recipe, we will add something that will make switching between languages much easier for users.

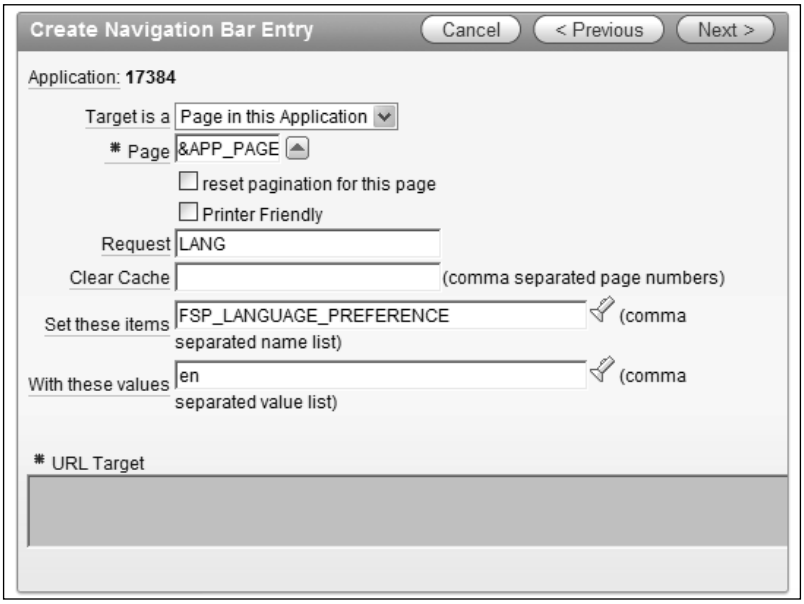
Getting ready

To start this recipe, we will need an application that has been translated like the application we created in the past two recipes.

How to do it...

The best place to create a language switch is a place that is visible at all times. In this case, we will use the navigation bar. Since this bar is in the header of our application, it is visible on all pages; ideal for our purposes.

1. Navigate to the **Shared Components**.
2. In the **Navigation Section** find the **Navigation Bar Entries** and click it.
3. There should already be an entry called **Logout** in place.
4. Click the **Create** button.
5. Confirm that the option **From Scratch** is selected and click **Next**.
6. Select **Navigation to URL** and click **Next.**
7. Enter **English** as the Entry Label and click **Next**.
8. In the next page, enter the following for the properties:
   - **Target is a**: **Page in this application**
   - **Page**: **&APP_PAGE_ID.** (do not forget the dot at the end)
   - **Request**: **LANG**
   - **Set these items**: **FSP_LANGUAGE_PREFERENCE**
   - **With these values**: **en**



9. Click **Next**.
10. Click **Create**.

We have now created a link in the navigation bar that will direct users to the English part of the application. To create a link to the Dutch version of the application, repeat these steps but change the Entry Label to 'Nederlands' and set **With these values** to **nl** instead of **en**.

Remember to run the XLIFF translation steps again to make sure the new text is available in both languages.

When we now run the application, we can see that two entries are available in the navigation bar.

Clicking on **Nederlands** will show the application in Dutch and clicking on **English** will change it back to the English language.

Translating data in an application

Besides the application labels, there is more to translate in an application. Data for instance.

In this recipe, we will see an example of this. To accomplish this, we will use a built-in way of translating the session language that is new in APEX 4.0.

Getting ready

Start by creating some new database objects. Remember that this is a crude setup. In a production environment, this should be more elaborate.

First, we will create a copy of the `EMP` table, but with a change. The `JOB` column will now be called `JOB_NO` and it's content will reference to a `EMP_JOB_TITLES` table.

```
create table EMP_LANG
(
  empno    NUMBER(4) not null,
  ename    VARCHAR2(10),
  job_no   NUMBER,
  mgr      NUMBER(4),
  hiredate DATE,
```

```
  sal      NUMBER(7,2),
  comm     NUMBER(7,2),
  deptno   NUMBER(2)
);
```

[emp_lang.sql]

Also create the EMP_JOB_TITLES table. This table contains a LANGUAGE column that will hold the language in which the `job_title` is entered for that row.

This also means that `job_no` is not a unique column in this table, but `job_no` in combination with language is the unique key.

```
create table EMP_JOB_TITLES
(
  job_no    NUMBER,
  job_title VARCHAR2(32),
  language  VARCHAR2(10)
);
```

[emp_job_titles.sql]

Next, it will need data. First the EMP_JOB_TITLES. This script will fill the table with the original 5 job titles from the EMP table with language 'en' and add 5 translations in Dutch for the same titles.

```
insert into EMP_JOB_TITLES (job_no, job_title, language)
values (1, 'PRESIDENT', 'en');
insert into EMP_JOB_TITLES (job_no, job_title, language)
values (2, 'MANAGER', 'en');
insert into EMP_JOB_TITLES (job_no, job_title, language)
values (3, 'ANALYST', 'en');
insert into EMP_JOB_TITLES (job_no, job_title, language)
values (4, 'CLERK', 'en');
insert into EMP_JOB_TITLES (job_no, job_title, language)
values (5, 'SALESMAN', 'en');
insert into EMP_JOB_TITLES (job_no, job_title, language)
values (1, 'Directeur', 'nl');
insert into EMP_JOB_TITLES (job_no, job_title, language)
values (2, 'Manager', 'nl');
insert into EMP_JOB_TITLES (job_no, job_title, language)
values (3, 'Analist', 'nl');
insert into EMP_JOB_TITLES (job_no, job_title, language)
values (4, 'Klerk', 'nl');
insert into EMP_JOB_TITLES (job_no, job_title, language)
values (5, 'Verkoper', 'nl');
```

[emp_job_titles_data.sql]

For the `EMP_LANG` table, copy the data from the `EMP` table, but replace the contents of the `JOB` column with the reference number as it was entered in the `EMP_JOB_TITLES` table (for example; `PRESIDENT` will become 1, `CLERK` will become 4).

Some examples of this data are:

```
insert into EMP_LANG (empno, ename, job_no, mgr, hiredate, sal, comm,
deptno)
values (7839, 'KING', 1, null, to_date('17-11-1981', 'dd-mm-yyyy'),
5000, null, 10);
insert into EMP_LANG (empno, ename, job_no, mgr, hiredate, sal, comm,
deptno)
values (7698, 'BLAKE', 2, 7839, to_date('01-05-1981', 'dd-mm-yyyy'),
2850, null, 30);
```

[emp_lang_data.sql]

The `emp_lang_data.sql` script contains 12 more rows.

How to do it

When there is a datamodel available that allows data to be saved in multiple languages, the hardest part is writing smart queries to get this data out in the right language for the user. We will create a page based on such a query for our simple datamodel.

The first step is getting the application ready. To use the **Session** option, take the following steps.

1. Go to the **Application Properties**.
2. Go to the **Globalization** tab.
3. In the select list **Application Language Derived From** select **Session**.
4. Click **Apply Changes**.

   Next step is to create a page based on a language-driven query.
5. On the application overview, click **Create Page**.
6. Select **Report** and click **Next**.
7. Select **Classic Report** and click **Next**.
8. Enter a number and name for the page and click **Next**.
9. Select **Do Not Use Tabs** and click **Next**.

10. Enter the query that will drive the Page.

```
select emp.empno
, emp.ename
, job.job_title
, emp.hiredate
, emp.sal
from emp_lang emp
, emp_job_titles job
where emp.job_no = job.job_no
and upper(job.language) = upper(apex_util.get_session_lang)
```

11. Click **Next** a few times until the **Finish** button appears and click it.

To test the changes, Run the page by altering the URL to accept a p_lang parameter like so:

```
http://server:port/apex/f?p=app_id:page_id:session&p_lang=en
```

Empno	Ename	Job Title	Hiredate	Sal
7369	SMITH	CLERK	17-DEC-80	1900
7499	ALLEN	SALESMAN	20-FEB-81	1600
7521	WARD	SALESMAN	22-FEB-81	1250
7566	JONES	MANAGER	02-APR-81	2975
7654	MARTIN	SALESMAN	28-SEP-81	1250
7698	BLAKE	MANAGER	01-MAY-81	2850
7782	CLARK	MANAGER	09-JUN-81	2450
7788	SCOTT	ANALYST	09-DEC-82	3200
7839	KING	PRESIDENT	17-NOV-81	5000
7844	TURNER	SALESMAN	08-SEP-81	1500
7876	ADAMS	CLERK	12-JAN-83	1100
7900	JAMES	CLERK	03-DEC-81	950
7902	FORD	ANALYST	03-DEC-81	3000
7934	MILLER	CLERK	23-JAN-82	1300

For the Dutch language, enter the URL like this:

```
http://server:port/apex/f?p=app_id:page_id:session&p_lang=nl
```

Empno	Ename	Job Title	Hiredate	Sal
7369	SMITH	Klerk	17-12-80	1900
7499	ALLEN	Verkoper	20-02-81	1600
7521	WARD	Verkoper	22-02-81	1250
7566	JONES	Manager	02-04-81	2975
7654	MARTIN	Verkoper	28-09-81	1250
7698	BLAKE	Manager	01-05-81	2850
7782	CLARK	Manager	09-06-81	2450
7788	SCOTT	Analist	09-12-82	3200
7839	KING	Directeur	17-11-81	5000
7844	TURNER	Verkoper	08-09-81	1500
7876	ADAMS	Klerk	12-01-83	1100
7900	JAMES	Klerk	03-12-81	950
7902	FORD	Analist	03-12-81	3000
7934	MILLER	Klerk	23-01-82	1300

How it works

Using the **Session** option for Globalization, allows developers to take advantage of some additional built-in functions and procedures.

1. `apex_util.set_session_lang(p_lang in varchar2)`
   - This procedure will set the language for the session to the value in the parameter
2. `apex_util.get_session_lang`
   - This function will get the current session language. It can also be called using the variable v('BROWSER_LANGUAGE')
3. `apex_util.reset_session_lang`
   - This procedure will clear the session language

In this recipe we took advantage of the apex_util.get_session_lang procedure to retrieve the correct job titles in our report. It doesn't take a lot of imagination to see the possibilities of this kind of construction. We have now seen how to put translatable data into a separate table and making it unique by combining an ID column like the JOB_NO column in our example to a LANGUAGE column.

Of course this can be used for any kind of data. Just remember to keep a keen eye on the performance of your application.

Where to buy this book

You can buy Oracle APEX 4.0 Cookbook from the Packt Publishing website:
`https://www.packtpub.com/oracle-apex-4-0-cookbook/book`

Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

www.PacktPub.com

For More Information:
www.packtpub.com/oracle-apex-4-0-cookbook/book