**Supporting Multiple Languages in Django**

## Internationalization vs Localization

Internationalization and localization represent two sides to the same coin. Together, they allow you to deliver your web application's content to different locales.

* **Internationalization**, represented by i18n (18 is the number of letters between i and n), is the processing of developing your application so that it can be used by different locales. This process is generally handled by developers.
* **Localization**, represented by l10n (10 is the number of letters between l and n), on the other hand, is the process of translating your application to a particular language and locale. This is generally handled by translators.

For more details: <https://www.w3.org/International/questions/qa-i18n>

eman@django\_locallibrary:sudo apt-get install gettext

eman@django\_locallibrary:sudo apt-get install gettext\_lazy

For the [LANGUAGE\_CODE](https://docs.djangoproject.com/en/3.2/ref/settings/#std:setting-LANGUAGE_CODE) to take effect, [USE\_I18N](https://docs.djangoproject.com/en/3.2/ref/settings/#std:setting-USE_I18N) must be True, which enables Django’s translation system.in setting.py add below:

# Internationalization

# https://docs.djangoproject.com/en/3.2/topics/i18n/

# Localization setting:

# LANGUAGE\_CODE = 'en-us'

LANGUAGE\_CODE = 'en'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

'UTC' is the default [TIME\_ZONE](https://docs.djangoproject.com/en/3.2/ref/settings/#std:setting-TIME_ZONE).

Since [USE\_L10N](https://docs.djangoproject.com/en/3.2/ref/settings/#std:setting-USE_L10N) is set to True, Django will display numbers and dates using the format of the current locale.

Finally, when [USE\_TZ](https://docs.djangoproject.com/en/3.2/ref/settings/#std:setting-USE_TZ) is True, datetimes will be time zone-aware.

Then add languages you want to add in setting.py:

from django.utils.translation import gettext, ngettext

from django.utils.translation import gettext\_lazy as \_

LANGUAGES = (

    ('en', \_('English')),

    ('fr', \_('French')),

    ('ar', \_('Arabic')),

)

See [list of language identifiers](http://www.i18nguy.com/unicode/language-identifiers.html) for more. <http://www.i18nguy.com/unicode/language-identifiers.html>

Add django.middleware.locale.LocaleMiddleware to the MIDDLEWARE settings list. This middleware should come after the SessionMiddleware because the LocaleMiddleware needs to use the session data. It should also be placed before the CommonMiddleware because the CommonMiddleware needs the active language to resolve the URLs being requested. Hence, the order is very important.Then it will be like below:

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

  'whitenoise.middleware.WhiteNoiseMiddleware',

  'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.locale.LocaleMiddleware', # localization and translation

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

Add a locale path directory in setting.py for your application where message files will reside:

LOCALE\_PATHS = [

    BASE\_DIR / 'locale/',

]

You need to create the "locale" directory inside of your root project and add a new folder for each language:

locale

├── en

├── ar

└── fr

Open the shell and run the following command from your project directory to create a .po message file for each language:

(env)$ django-admin makemessages --all --ignore=env

You should now have:

locale

├── en

│ └── LC\_MESSAGES

│ └── django.po

├── ar

│ └── LC\_MESSAGES

│ └── django.po

└── fr

└── LC\_MESSAGES

└── django.po

Take note of one of the .po message files:

1. msgid: represents the translation string as it appears in the source code.
2. msgstr: represents the language translation, which is empty by default. You'll have to supply the actual translation for any given string.

Currently, only the LANGUAGES from our settings.py file have been marked for translation. Therefore, for each msgstr under the "fr" and "es" directories, enter the French or Spanish equivalent of the word manually, respectively. You can edit .po files from your regular code editor; however, it's recommended to use an editor designed specifically for .po like [Poedit](https://poedit.net/).

Translating Templates, Models, and Forms

You can translate model field names and forms by marking them for translation using either the gettext or gettext\_lazy function:

In each file you want to translate is need to import gettext\_lazy function first

from django.utils.translation import gettext\_lazy as \_

and make each word need to translate as below: add \_ (‘ word or paragraph‘) in setting file as below:

def my\_view(request):

    output = \_("Welcome to my site.")

    return HttpResponse(output)

You have to add **{% load i18n %}** at the top of the HTML file to use the translation templates tags.Add below in the top of each html file need to be translated:

{% load i18n %}

<h1>{% trans "hello this is Home page of book list view" %}</h1>

<p>{% trans "hello this is Home page of book list view" %}</p>

Don't forget to add {% load i18n %} to the top of the file.

(env)$ django-admin makemessages --all --ignore=env

Update the following msgstr translations:

In .po files for each language you added.

Compile the messages:

(env)$ django-admin compilemessages --ignore=env

Or we can Using Rosetta Translation Interface:

Rosetta is a [Django](http://www.djangoproject.com/) application that facilitates the translation process of your Django projects.

<https://django-rosetta.readthedocs.io/>

pip3 install django-rosetta

edit setting.py as below:

# Application definition

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',#Core authentication framework and its default models.

    'django.contrib.contenttypes',#Django content type system

(allows permissions to be associated with models).

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

     # Add our new application

    'catalog.apps.CatalogConfig',#This object was created for us in /catalog/apps.py

    'rosetta', # NEW to tranlsator

]

You'll also need to add Rosetta's URL to your main URL configuration in project/urls.py:

urlpatterns = [

    path('admin/', admin.site.urls),

    path('catalog/', include('catalog.urls')),

    path('', views.index, name='index'),

    path('', RedirectView.as\_view(url='catalog/', permanent=True)),

    #Add Django site authentication urls (for login, logout, password management)

    path('accounts/', include('django.contrib.auth.urls')),

    path('rosetta/', include('rosetta.urls')),  # NEW

]+ static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)

Create and apply the migrations, and then run the server:

(env)$ python manage.py makemigrations

(env)$ python manage.py migrate

(env)$ python manage.py runserver

Make sure you're logged in as an admin, and then navigate to <http://127.0.0.1:8000/rosetta/> in your browser:When you finish editing translations, click the "Save and translate next block" button to save the translations to their respective .po file. Rosetta will then compile the message file, so there's no need to manually run the

django-admin compilemessages --ignore=env command.

Note that after you add new translations in a production environment, you'll have to reload your server after running the django-admin compilemessages --ignore=env command, or after saving the translations with Rosetta, for changes to take effect.

Add Language Prefix to URLs:

Next, add the i18n\_patterns function to project/urls.py:

from django.conf.urls.i18n import i18n\_patterns

from django.utils.translation import gettext\_lazy as \_

urlpatterns = i18n\_patterns(

    path('admin/', admin.site.urls),

    path('catalog/', include('catalog.urls')),

    path('', views.index, name='index'),

    path('', RedirectView.as\_view(url='catalog/', permanent=True)),

    #Add Django site authentication urls (for login, logout, password management)

    path('accounts/', include('django.contrib.auth.urls')),

    path('rosetta/', include('rosetta.urls')),  # NEW

)+ static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)

Run the development server again, and navigate to <http://127.0.0.1:8000/> in your browser. You will be redirected to the requested URL, with the appropriate language prefix. Take a look at the URL in your browser; it should now look like <http://127.0.0.1:8000/en/>.

Change the requested URL from en to either fr or es. The heading should change.

**Allowing Users to Switch Languages**

Update the index.html or base\_generic.html file like so:

{% load i18n %}

<!DOCTYPE html>

<html lang="en">

<head>

  {% block title %}<title>First Home Page</title>{% endblock %}

  <meta charset="utf-8">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css" integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2" crossorigin="anonymous">

  <div class="container-fluid">

    <div class="row">

      <div class="col-sm-2">

      {% block sidebar %}

        <ul class="sidebar-nav">

      {% get\_current\_language as CURRENT\_LANGUAGE %}

      {% get\_available\_languages as AVAILABLE\_LANGUAGES %}

      {% get\_language\_info\_list for AVAILABLE\_LANGUAGES as languages %}

      <div class="languages">

        <p>{% trans "Language" %}:</p>

        <ul class="languages">

          {% for language in languages %}

            <li>

              <a href="/{{ language.code }}/"

          {% if language.code == CURRENT\_LANGUAGE %} class="active"{% endif %}>

                {{ language.name\_local }}

              </a>

            </li>

          {% endfor %}

        </ul>

      </div>