Archived:Localization Example for PySymbian

Archived: This article is **archived** because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

Introduction

01 Feb 2009

This article presents a strategy for supporting multiple languages in a Python application for S60 devices. Since the default language is defined, additional translations may be added whenever needed. Moreover, missing translations are replaced by the default translation, allowing incremental translations without breaking the code.

The strategy is composed of a main script file (**wm_locale.py**) used for dynamic loading of the desired language, and for localisation files that contain the translations. The localisation files are also Python files and they are imported as modules. Using Python introspection, the translation may be loaded and missing translations are replaced by the default language.

Code

The main script is below.

wm_locale.py

```
# -*- coding: utf-8 -*-
#
# Marcelo Barros de Almeida
# marcelobarrosalmeida (at) gmail.com
#

__all__ = [ "Locale" ]

class Loc_Data(object):
    "Translation data holder"
    pass

class Default(object):
    "Default language support"
    def __init__(self):
        self.loc = Loc_Data()
        self.loc.zero = u'Zero'
        self.loc.one = u'One'
        self.loc.two = u'Two'
        self.loc.three = u'Three'
        self.loc.four = u'Four'
        self.loc.five = u'Five'
        self.loc.six = u'Six'
        self.loc.seven = u'Seven'
        self.loc.eight = u'Eight'
        self.loc.nine = u'Nine'
        self.loc.change_language = u'Change Language'
        self.loc.english_us = u'English (USA)'
        self.loc.finnish = u'Finnish'
        self.loc.hungarian = u'Hungarian'
        self.loc.portuguese_br = u'Portuguese (Brazil)'
        self.loc.about = u'About'
        self.loc.exit = u'Exit'
```


(C) Copyright Nokia 2014. All rights reserved.

```
class Locale(Default):
    "Multiple language support class"

    LOC_MODULE = "wm_locale_%s"

    def __init__(self,lang = ""):
        "Load all locale strings for one specific language or default if empty"
        self.set_locale(lang)

    def set_locale(self,lang = ""):
        "Load all locale strings for one specific language or default if empty"
        Default.__init__(self)

        try:
            lang_mod = __import__( self.LOC_MODULE % lang )
        except ImportError:
            pass
        else:
            self.merge_locale(lang_mod)

    def merge_locale(self, lang_mod):
        "Merge new location string into default locale"

        # replace existing strings and keep old ones
        # if it is missing in the locale module
        for k,v in self.loc.__dict__.iteritems():
            if hasattr(lang_mod,k):
                nv = getattr(lang_mod, k)
                setattr(self.loc,k,nv)
```

All default translations are defined in class **Default()** using the attribute **self.loc**. Each string in your program should be represented by a different attribute in self.loc.

Localisation modules are loaded dynamically and the files are named according to certain conventions. This is represented by the following expression:

```
LOC_MODULE = "wm_locale_%s"
```

So, if you have a **pt_BR** translation, create a file called **wm_locale_pt_BR.py**. Inside this module, translate all strings in class **Default()**, removing any class or **self.loc** references. For instance, the pt_BR translation would be:

wm_locale_pt_BR.py

```
# -*- coding: utf-8 -*-
#
# Marcelo Barros de Almeida
# marcelobarrosalmeida (at) gmail.com
#
zero = u'Zero'
one = u'Um'
two = u'Dois'
three = u'Três'
four = u'Quatro'
five = u'Cinco'
six = u'Seis'
```


(C) Copyright Nokia 2014. All rights reserved.

```
seven = u'Sete'
eight = u'Oito'
nine = u'Nove'
change_language = u'Mudar idioma'
english_us = u'Inglês (EUA)'
finnish = u'Finlandês'
hungarian = u'Húngaro'
portuguese_br = u'Português (Brasil)'
about = u'Sobre'
exit = u'Sair'
```

The next code snippet demonstrates how the locale class can be used.

wm_locale_demo.py

```
# -*- coding: utf-8 -*-
#
# Marcelo Barros de Almeida
# marcelobarrosalmeida (at) gmail.com
#

import sys
sys.path.append(r'e:\python')

import appuifw
import e32
import wm_locale

class Locale_Demo(object):
    def __init__(self):
        appuifw.app.exit_key_handler = self.close
        appuifw.app.title = u"Locale Demo"
        self.update_locale()
        self.app_lock = e32.Ao_lock()

    def close(self):
        self.app_lock.signal()

    def about(self):
        appuifw.note( u"Locale Demo by Marcelo Barros", "info" )

    def update_locale(self,lang=""):
        self.labels = wm_locale.Locale(lang)
        self.refresh()

    def refresh(self):
        entries = [
            self.labels.loc.zero,
            self.labels.loc.one,
            self.labels.loc.two,
            self.labels.loc.three,
            self.labels.loc.four,
            self.labels.loc.five,
            self.labels.loc.six,
            self.labels.loc.seven,
            self.labels.loc.eight,
```


(C) Copyright Nokia 2014. All rights reserved.

```
            self.labels.loc.nine
            ]

        self.body = appuifw.Listbox(entries)

        self.menu = [
            (self.labels.loc.change_language, (
                (self.labels.loc.english_us, lambda: self.update_locale("en_US")),
                (self.labels.loc.finnish, lambda: self.update_locale("fi")),
                (self.labels.loc.hungarian, lambda: self.update_locale("hu")),
                (self.labels.loc.portuguese_br, lambda: self.update_locale("pt_BR"))
                )
             ),
            (self.labels.loc.about, self.about),
            (self.labels.loc.exit, self.close)
            ]

        appuifw.app.menu = self.menu
        appuifw.app.body = self.body

    def run(self):
        self.app_lock.wait()
        appuifw.app.menu = []
        appuifw.app.body = None
        appuifw.app.set_exit()

if __name__ == "__main__":

    ld = Locale_Demo()
    ld.run()
```

If you have more translation files, just add them to your project. Translations below were provided by [1] and you can see that there are missing translations.

wm_locale_en_US.py

```
zero = u'Zero'
one = u'One'
two = u'Two'
three = u'Three'
four = u'Four'
five = u'Five'
six = u'Six'
seven = u'Seven'
eight = u'Eight'
nine = u'Nine'
change_language = u'Change Language'
english_us = u'English (USA)'
finnish = u'Finnish'
hungarian = u'Hungarian'
portuguese_br = u'Portuguese (Brazil)'
about = u'About'
exit = u'Exit'
```

wm_locale_fi.py

(C) Copyright Nokia 2014. All rights reserved.

```
zero = u'nolla'
one = u'yksi'
two = u'kaksi'
three = u'kolme'
four = u'neljä'
five = u'viisi'
six = u'kuusi'
seven = u'seitsemän'
eight = u'kandeksan'
nine = u'yhdeksän'
change_language = u'Vaihda kieli'
english_us = u'englanti'
finnish = u'suomi'
hungarian = u'unkari'
about = u'Tietoja'
exit = u'Poistu'
```

wm_locale_hu.py

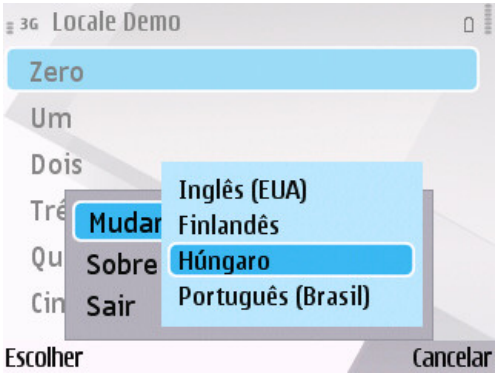
```
zero = u'nulla'
one = u'egy'
two = u'kett\u0151'
three = u'három'
four = u'négy'
five = u'öt'
six = u'hat'
seven = u'hét'
eight = u'nyolc'
nine = u'kilenc'
change_language = u'Nyelv változtatás'
english_us = u'angol'
finnish = u'finn'
hungarian = u'magyar'
about = u'Információ'
exit = u'Kijárat'
```

Screenshots

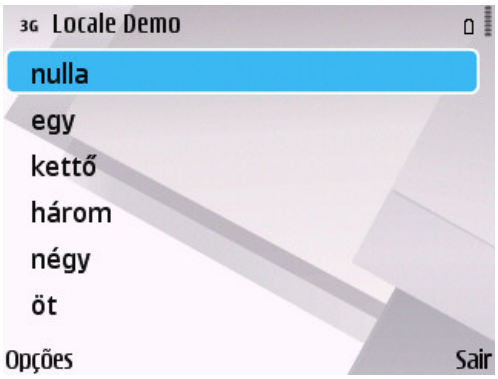
Locale demo showing listbox elements in English and language selection menu.

Locale demo showing listbox elements in Portuguese and language selection menu.

(C) Copyright Nokia 2014. All rights reserved.



Locale demo showing listbox elements in Hungarian.



Source Code

Donwload locale demo source code: MBA locale demo src.zip

References

Thanks to Rafael T. and JOM for fruitful discussions on the Python discussion board.

- Localization Test 1.00
- Strategy for supporting multiple languages

This article is also available in Portuguese,

- Adicionando suporte a várias linguagens em PySymbian

(C) Copyright Nokia 2014. All rights reserved.