

ТЕМА 1. МОДУЛИ QT ДЛЯ СОЗДАНИЯ ПРИЛОЖЕНИЙ С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ.

ЛЕКЦИЯ 3. ЛОКАЛИЗАЦИЯ ПРИЛОЖЕНИЯ.



УЧЕБНЫЕ ВОПРОСЫ

- 1. Понятие локализации
- 2. Локализация в Qt (PySide2).

1. ПОНЯТИЕ ЛОКАЛИЗАЦИИ

- Локализация программного обеспечения — процесс адаптации программного обеспечения к культуре какой-либо страны.
- В практике написания программного обеспечения (далее - ПО) - перевод пользовательского интерфейса, документации и сопутствующих файлов программного обеспечения с одного языка на другой.
- Пример:

Имя	<input type="text"/>
Фамилия	<input type="text"/>
<input type="button" value="Ввод"/>	



Name	<input type="text"/>
Surname	<input type="text"/>
<input type="button" value="Input"/>	

1.1. ЛОКАЛИЗАЦИЯ В ШИРОКОМ СМЫСЛЕ

- 1. Изучение целевой аудитории
- 2. Изучение культурных и религиозных особенностей
- 3. Локализация приложения

1.1.1. ИЗУЧЕНИЕ ЦЕЛЕВОЙ АУДИТОРИИ

- стоимость различных платных функций для разных стран и регионов может варьироваться;
- в некоторых странах платную функцию придётся сделать бесплатной, допустим и обратный процесс;
- бывают ситуации, когда цены могут варьироваться даже от платформ, на которых исполняется ваше приложение.

1.1.2. ИЗУЧЕНИЕ КУЛЬТУРНЫХ И РЕЛИГИОЗНЫХ ОСОБЕННОСТЕЙ

- В Иране очень негативно относятся к собакам, по религиозным соображениям и их количество в этой стране очень мало;
- В Мексике не принято дарить жёлтые розы, т.к. они символизируют смерть;

1.2.1 ЛОКАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

Основные шаги при локализации приложения:

1. Обеспечить поддержку языка и национальных стандартов.
2. Перевод текстов в интерфейсе программы на целевой язык.
3. Тонкая настройка под целевую страну.

1.2.2. ОБЕСПЕЧЕНИЕ ПОДДЕРЖКИ ЯЗЫКА И НАЦИОНАЛЬНЫХ СТАНДАРТОВ.

- Соответствие товарным знакам целевой страны (документация к ПО, антимонопольное законодательство, законы о хранении персональных данных);
- Адаптация приложения под шрифты целевой страны;
- Соответствие стандартам целевой страны (Формат даты, времени, дробных и многозначных чисел, система мер и весов, форматы бумаги)

1.2.3. ПЕРЕВОД ТЕКСТОВ В ИНТЕРФЕЙСЕ ПРОГРАММЫ НА ЦЕЛЕВОЙ ЯЗЫК.

- В сложном ПО не все части стоит переводить (название функций Excel, MySQL, ошибок вашего приложения);
- Корректная расстановка (подгонка) элементов интерфейса в соответствии с особенностями языка целевой страны (существуют языки с написанием справа налево (арабский, иврит) и сверху вниз (японский));
- Адаптация изображений, звуков для целевой страны (текст, возможная замена)

1.2.4. ТОНКАЯ НАСТРОЙКА ПОД ЦЕЛЕВУЮ СТРАНУ.

- Работа со словоформами («найдено 4 файлов»)
- Доп. стандарты, не влияющие на функциональность (формат даты/времени)
- Взаимодействие с другим ПО (бухгалтерские программы разные для разных стран)
- Учёт менталитета (цвета, юмор, пасхалки)
- Перерисовка графики (дорожные знаки, розетки, флаги)

2. ЛОКАЛИЗАЦИЯ В PYSIDE2

- Для локализации и интернационализации приложений в PySide2 используется класс `QTranslator`.
- Из официальной документации:

`QtCore.QTranslator`

“””Объект этого класса содержит набор переводов с исходного языка на целевой язык. `QTranslator` предоставляет функции для поиска переводов в файле перевода. Файлы перевода создаются с помощью Qt Linguist.

Чаще всего `QTranslator` используется для загрузки файла перевода и его установки с помощью `QtCore.QCoreApplication.installTranslator()`“””

2.1. ИНСТРУМЕНТЫ ЛОКАЛИЗАЦИИ PYSIDE2 ПРИЛОЖЕНИЯ

- PySide2 обеспечивает отличную поддержку для перевода приложений на целевые языки.
- «*lupdate*» - используется для синхронизации исходного кода и переводов.
- «*lrelease*» - используется для создания файлов перевода в приложении.
- «*Qt Linguist*» - инструмент для переводчиков используется для перевода слов/фраз в файлах созданных с помощью инструмента *lupdate*.
- Более подробное описание инструментов в документации.

2.2. ИСПОЛЬЗОВАНИЕ ИНСТРУМЕНТОВ ЛОКАЛИЗАЦИИ

- 1. Создаём приложение.

```
1 import sys
2 from functools import partial
3 from PySide2 import QtCore, QtWidgets
4
5 class MyTranslateApp(QtWidgets.QMainWindow):
6     |
7     def __init__(self):
8         super().__init__()
9
10
11 if __name__ == "__main__":
12     app = QtWidgets.QApplication(sys.argv)
13     myapp = MyTranslateApp()
14     myapp.show()
15     sys.exit(app.exec_())
```

- 2. Создаём объект класса QTranslator

```
# Инициализируем класс QTranslator, который потребуется для
# локализации пользовательского текста в самой программе
self.translator = QtCore.QTranslator(self)
```

- 3. Добавляем текст через «конструктор»

```
self.labelName.setText(QtWidgets.QApplication.translate("MyTranslateApp", "Имя"))
```

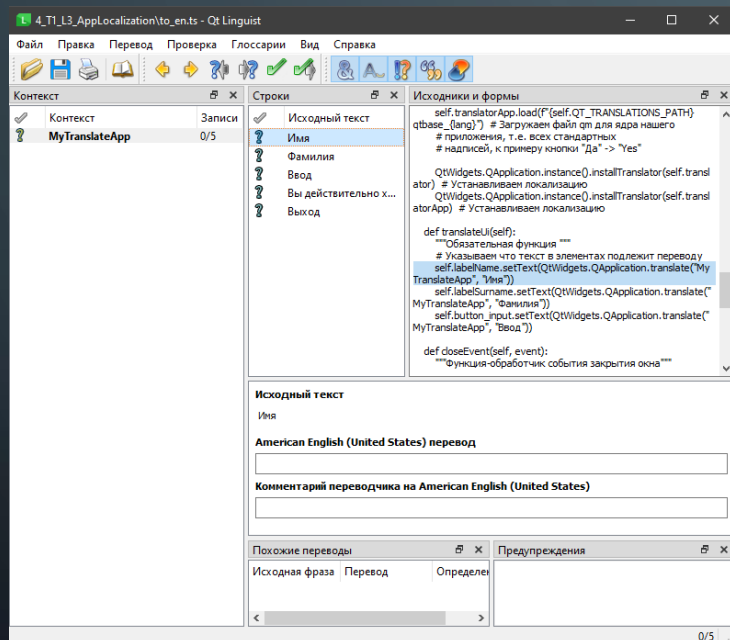
- 4. Создаём .ts-файл

```
lupdate MyLocalizationApp.py -ts to_en.ts
```

2.2. ИСПОЛЬЗОВАНИЕ ИНСТРУМЕНТОВ ЛОКАЛИЗАЦИИ (ПРОДОЛЖЕНИЕ)

5. Редактируем файл .ts

5.1. Через QtLinguist



5.2. Вручную (структура xml)

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE TS>
3 <TS version="2.1" language="en_US">
4 <context>
5   <name>MyTranslateApp</name>
6   <message>
7     <location filename="MyLocalizationApp.py" line="98"/>
8     <source>Имя</source>
9     <translation type="unfinished"></translation>
10  </message>
11  <message>
12    <location filename="MyLocalizationApp.py" line="99"/>
13    <source>Фамилия</source>
14    <translation type="unfinished"></translation>
15  </message>
16  <message>
17    <location filename="MyLocalizationApp.py" line="100"/>
18    <source>Ввод</source>
19    <translation type="unfinished"></translation>
20  </message>
21  <message>
22    <location filename="MyLocalizationApp.py" line="107"/>
23    <source>Вы действительно хотите закрыть программу?</source>
24    <translation type="unfinished"></translation>
25  </message>
26  <message>
27    <location filename="MyLocalizationApp.py" line="111"/>
28    <source>Выход</source>
29    <translation type="unfinished"></translation>
30  </message>
31 </context>
32 </TS>
```

2.2. ИСПОЛЬЗОВАНИЕ ИНСТРУМЕНТОВ ЛОКАЛИЗАЦИИ (ПРОДОЛЖЕНИЕ)

6. Конвертируем .ts в .qm

6.1. Через терминал

```
lrelease.exe to_en.ts to_en.qm
```

6.2. Через QtLinguist

«Файл» -> «Скомпилировать как...»

7. Загружаем файл .qm в программе

```
def setLocalization(self, lang):  
    self.translator.load(f"to_{lang}")
```

8. Устанавливаем локализацию

```
QtWidgets.QApplication.instance().installTranslator(self.translator)
```

9. Переопределяем метод changeEvent для обновления окна

```
def changeEvent(self, event):  
    if event.type() == QtCore.QEvent.LanguageChange: # Если "ловим" событие LanguageChange  
        self.translateUi() # Тогда вызываем функцию translateUi()  
    super(MyTranslateApp, self).changeEvent(event) # Отправляем "родителю" новое состояние
```


ИТОГИ:

- Локализация — это сложная и всеобъемлющая операция.
- Уже при разработке ПО соображения будущей интернационализации должны учитываться самым серьёзным образом. Мы привыкли видеть программное обеспечение, русифицированное по первому-второму уровню; сложного ПО с исчерпывающей русификацией практически не существует.
- Примером глубокой локализации может служить операционная система, где локализация нередко включает и национально-ориентированные пиктограммы.

Спасибо за внимание!