Aix*Marseille université
Initiative d'excellence
Année 2018-19

Diplôme Inter-Universitaire
Enseigner l'informatique au lycée

Bloc 1 : TP 7

Interfaces homme-machine

1 But de l'activité

L'application que nous allons développer permettra de se renseigner sur la météo dans une ville de la région PACA. Elle s'appuie sur l'utilisation de OpenWeatherMap, un service en ligne qui fournit des données météorologiques, notamment des données météorologiques actuelles, des prévisions et des données historiques, aux développeurs de services Web et d'applications mobiles, au travers d'une API python `pyowm` : `https://pyowm.readthedocs.io/en/latest/`

Au cours de cette activité, vous allez apprendre à générer une application munie d'une Interface Homme-Machine construite à l'aide de QtDesigner. Si vous avez installé Anaconda, vous avez déjà tout ce dont vous avez besoin. Sinon, vous devrez installer les paquets suivants :

```
$ sudo apt-get install python3-pyqt5
$ sudo apt-get install qttools5-dev-tools
```

ou télécharger PyQt5 depuis le site de Riverbankcomputing : `https://riverbankcomputing.com/software/pyqt/download5`

1.1 Depuis pycharm

1.1.1 Créer un nouveau projet

- Ajouter un environnement d'exécution au projet et lui ajouter les packages suivants :
  - PyQt5 : QtCore, QtWidgets, QtGUI
  - pyowm : API openWheatherMap
  - pytz : Gestion time zone

1.2 Depuis un autre éditeur

1.2.1 Ajout des packages

- pyowm : https ://pyowm.readthedocs.io/en/latest/#installation
- https ://pypi.org/project/pytz/

2 L'IHM

2.1 Réalisation

L'IHM est réalisée au moyen de QtDesigner. Depuis le terminal de pycharm (ou depuis un terminal de l'OS), exécuter la commande suivante :

```
$ designer
```

2.1.1 Le formulaire d'IHM

- Créer un nouveau formulaire (fenètre d'IHM) de type Main Window
- Glisser les widgets suivants sur le formulaire :
  - Groups Box

- Combo Box
- Push Button
- Label
- LCD Number

— Construiser une IHM qui devrait avoir cette allure :

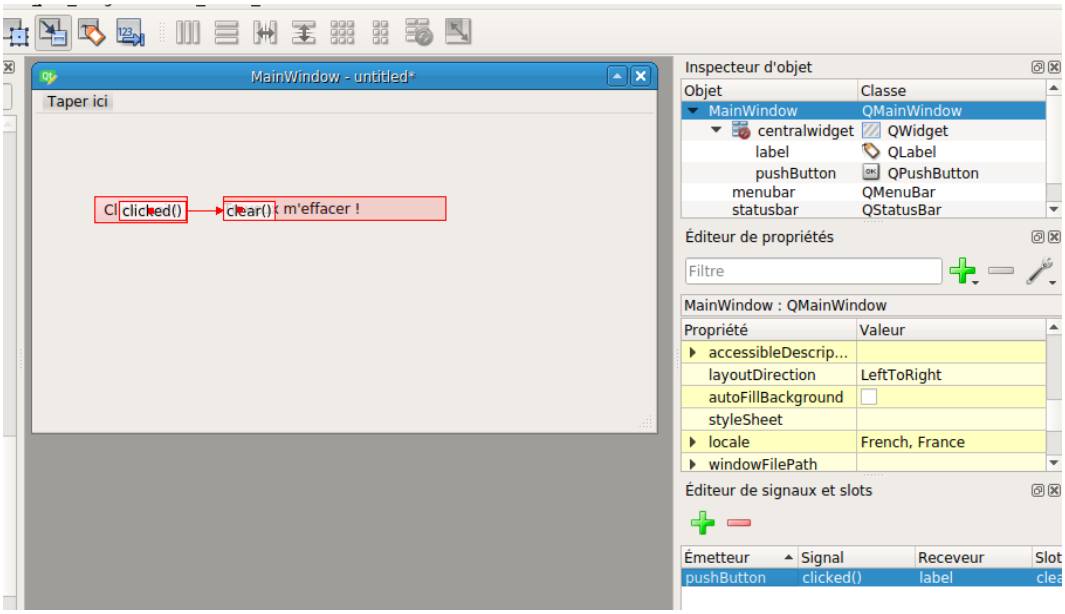
2.1.2 Nommage des widgets

Il est important de donner des noms cohérents qui permettent d'identifier facilement les widgets lors de l'écriture du code. Une convention généralement admise consiste à indiquer par les deux ou trois premières lettres en minuscules le type de widget :

widget	nom
Combo Box Département	cbDepartements
Combo Box Ville	cbVilles
Label Prévision	lbPrevisionsVille
Push Button Actualiser	pbActualiser
Label Date des données	lbDatePrevisions
LCD Number Température	lcdTemperature
LCD Number Humidité	lcdHumidity
LCD Number Pression	lcdPression
LCD Number Vitesse du vent	lcdVitesseVent
LCD Number Direction du vent	lcdDirectionVent
LCD Number Nuages	lcdNuages
LCD Number Pluie	lcdPluie
LCD Number Neige	lcdNeige

2.1.3 Edition des signaux/slots

— Cliquer sur l'icone `Editer signaux/slots` ou menu `Edition -> Editer signaux/slots`
— Cliquer sur le widget qui va émettre le signal et tirer pour créer le lien.
— Relâcher le lien dans le vide pour générer le slot à connecter



— Dans la fenètre **Configurer connexion** qui s'est ouverte, cliquer sur le signal `CurrentTextChanged(QString)` qui correspond à l'évènement que l'on souhaite gérer pour le widget `cbDepartement`
— Cliquer sur le bouton **Editer** de `Main Window` pour ajouter un nouveau slot
— Cliquer sur le bouton + de **Slots**
— Renommer le slot créé en `fill_cb_cities(QString)` : le QString en paramètre sera celui fourni par le signal.
— Procéder de même pour les autres signaux solts :

Emetteur	Signal	Slot
cbDepartements	currentTextChanged(QString)	fill_cb_city(QString)
cVilles	currentTextChanged(QString)	show_city()
pbActualiser	clicked()	previsions()

2.1.4 générer la classe Ui_MainWindow

— Enregistrer le travail sous le nom **weatherPaca.ui**. L'extension .ui désigne la description de l'IHM (user interface). C'est un fichier XML.

La description de l'interface graphique est générée au moyen de la commande `pyuic5`. Dans le terminal de pycharm, exécuter la commande suivante :

```
$ pyuic5 weatherPaca.ui -o weatherPacaGUI.py
```

Le fichier **weatherPacaGUI.py** (GUI : Graphic User Interface) contient le code python de la classe `Ui_MainWindow`. Il ne reste plus qu'à l'importer dans notre programme.

3 Le programme

— Créer un nouveau script python nommé **weatherPaca.py** dans le même dossier que le fichier **weatherPacaGui.py**.

3.1 Importer les modules :

```
from PyQt5 import QtWidgets
from weatherPacaGUI import Ui_MainWindow
import sys
import pyowm
from _datetime import datetime
import pytz
```

3.2 Code de l'application

3.2.1 Code la classe de l'IHM :

```
class ApplicationIHM(QtWidgets.QMainWindow):
    def __init__(self):
        super(ApplicationIHM, self).__init__()

        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

    def fill_cb_city(self, dept):
        print("fill_cb_city")
        # todo

    def show_city(self):
        print("show_city")
        # todo

    def previsions(self):
        print("previsions")
        # todo
```

3.2.2 Code de la fonction main()

```
def main():
    app = QtWidgets.QApplication(sys.argv)
    application = ApplicationIHM()
    application.show()
    sys.exit(app.exec_())

if __name__ == "__main__":
    main()
```

3.2.3 Premier test

L'application devrait pouvoir s'exécuter. En revanche, elle ne fait strictement rien étant donné que nous n'avons pas décrit les actions à mener dans les slots.

3.2.4 Code du slot fill_cb_city

Ce slot à pour but de charger la combo box `cbVilles` avec le contenu du fichier correspondant au département qui aura été choisi dans la combo box `cbDepartements`.

Nous allons donc commencer par remplir la combo box `cbDepartements` avec le contenu du fichier `paca/paca.txt`. Cette action doit être faite dès le lancement de l'application. C'est donc dans le constructeur qu'elle doit être écrite.

```
...
self.ui.setupUi(self)

f = open("paca/paca.txt", 'r')
lines = f.readlines()
f.close()
lines_stripped = []
for dep in lines:
    lines_stripped.append(dep.strip("\n"))
print(lines)
print(lines_stripped)
self.ui.cbDepartements.insertItems(0, lines_stripped)
self.ui.cbDepartements.setCurrentText(lines_stripped[0])
```

— Tester le remplissage de la combo box `cbDepartements`.

Pour remplir correctement la combo box `cbVilles`, il faut choisir le bon fichier en fonction du département choisi :

— Alpes de hautes Provence : `paca/alpesdehauteprovence.txt`
— Hautes Alpes : `paca/hautesalpes.txt`
— Alpes Maritimes : `paca/alpesmaritimes.txt`
— Bouches du Rhône : `paca/bouchesdurhone.txt`
— Vaucluse : `paca/vaucluse.txt`

— A partir du code de remplissage de la combo box `cbDepartements`, écrire le code du slot `fill_cb_city(self, dept)` ou `dept` est le nom du département transmis par le signal `currentTextChanged(QString)` émis par `cbDepartements` à destination du slot `fill_cb_city()`

3.2.5 Code du slot show_city

Lorsqu'une ville est sélectionnée dans la combo box `cbVilles`, on affiche son nom dans le label `lbPrevisionsVille` au moyen de la méthode `setText(QString)` :

```
self.ui.lbPrevisionsVille.setText("Prévisions pour : "+self.ville)
```

Remarque :

— un widget de l'IHM est un membre de la classe `Ui_MainWindow`, accessible depuis l'instance de cette classe `ui`, lui même membre de la classe courante.
— l'objet `ville` qui contiendra le nom de la ville sélectionnée dans `cbVille` est précédé de `self`. C'est donc un membre de la classe courante et il a donc une portée globale sur toute la classe. Sans ce mot clé, un objet ou une variable à une portée locale limité à la méthode ou à la fonction dans laquelle il est utilisé.
— Pour obtenir le texte de l'item sélectionné dans une combo box, on utilise la méthode `currentText()`.

— Ecrire le code du slot `show_city()` pour qu'il affiche le nom de la ville sélectionnée dans la combo box `cbVilles` dans le label `lbPrevision`, précédé de "Prévision pour".

Lorsque la ville est choisie il faut afficher les prévisions. C'est le comportement du slot `prevision()`. Il suffira donc de l'invoquer.

3.3 Accès aux données météo

Les données météo nous serons fournies par openWeatherMap au travers de leur API `pyowm`. Une API (Application Programming Interface) est un ensemble de classes, de méthodes ou de fonctions qui permettent de réaliser une interface entre un programme et un service tiers. Pour bénéficier des services de l'API d'openWeatherMap, il faut s'inscrire et obtenir une clé d'API :

`https://openweathermap.org/appid`

L'activation de la clé peut prendre quelques heures, aussi, il serait bon que vous vous y preniez un peu à l'avance. Vous pourrez toutefois utiliser une de mes clés, la seule restriction qui pourrait poser problème est que le nombre d'utilisation est limité à 60 par minutes.

La clé gratuite permet d'avoir des prévisions à 5 jours depuis n'importe quel endroit ou ville. Les données météorologiques sont actualisées toutes les 3 heures. La prévision est fournie au format JSON ou XML.

Le module python `pyown` exploite cette API et fourni un ensemble d'objets simple et convivial pour accéder au données d'openWeatherMap : `https://pyowm.readthedocs.io/en/latest/`

3.3.1 Connexion à l'API

— Identifier dans la documentation de `pyowm` l'objet à utiliser pour établir une connexion avec l'API.

On utilisera cet objet dans le constructeur de notre classe :

```
class ApplicationIHM(QtWidgets.QMainWindow):
    def __init__(self):
        super(ApplicationIHM, self).__init__()

        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

        ...

        print("Connexion à OpenWheatherMap")
        self.owm = pyowm.OWM('d24df2ad820eeec286b0b612714a6a96')
        # ma clé d'API que vous pouvez utiliser (avec parcimonie)
```

Rappel : l'objet `own` est précédé du mot clé `self`. On peut l'utiliser partout dans la classe.

3.3.2 La météo actuellement observée dans la ville choisie

— Identifier dans la documentation de `pyowm` la méthode à appliquer à l'objet `own` pour obtenir la météo actuelle à un emplacement spécifique. L'application de cette méthode retourne un objet d'observation que l'on notera `observation`. Il n'a pas besoin d'avoir une porté global car on ne l'utilisera que dans le slot `previsions()`. Il ne sera donc pas précédé du mot clé `self`.
— Compléter le slot `previsions()`

```
def previsions(self):
    observation = self.owm.weather_at_place(str(self.ville))
```

3.3.3 Obtenir les données météo

— Identifier dans la documentation de `pyowm` la méthode à appliquer à l'objet `observation` pour obtenir les données météo. L'application de cette méthode retourne un objet météo que l'on notera `weather`.
— Compléter le slot `previsions()`

```
observation = self.owm.weather_at_place(str(self.ville))
weather = observation.get_weather()
```

On peut désormais accéder à la température, le taux d'humidité, la pression atmosphérique, la couverture nuageuse, le volume de pluie et de neige tombé dans les trois heures qui ont précédé le dernier envoi.

On accède également à la date et l'heure de référence des données. Il s'agit du l'instant où les données météo ont été publiées. C'est une date UTC, il nous faut la rapporter à notre fuseau horaire :
— Compléter le slot `previsions()`

```
now = weather.get_reference_time()
tz = pytz.timezone('Europe/Paris')
print(f'Timestamp prévisions : {now}')
la_date = datetime.fromtimestamp(now,tz).strftime('%d/%m/%Y - %H:%M:%S')
print(f'Date prévisions : {la_date}')
self.ui.lbDatePrevisions.setText(str(la_date))
```

Les données météo sont prévues pour être affichées dans des widgets `lcdNumber`. On affiche une valeur dans ces widget par l'application de leur méthode `display(number)`.
— Identifier les méthodes à appliquer à l'objet `weather` pour obtenir les différentes données météo et utiliser la valeur de retour pour la passer en paramètre de la métode `diplay(number)` du bon `lcdNumber`.
— Compléter le slot `previsions()`

```
humidity = weather.get_humidity()
print(humidity)
self.ui.lcdHumidity.display(humidity)
```

Remarque : Attention aux informations retournées. Ce n'est pas toujours si simple !

4 Aller plus loin

— Explorer la documentation de `pyowm` chercher les méthodes à utiliser pour obtenir les prévisions à 5 jours
— Utiliser la classe `QTimer` pour automatiser le rafraichissement des données
— Ajouter un widget `label` pour afficher une image relative aux conditions météo (soleil, nuages, pluie, neige, ...)
— Sauvegarder les données dans un fichier csv
— ...