QApplication

QApplicationは、PySide6アプリケーション全体を管理するクラスです。

アプリケーション内でこのクラスのインスタンスは1つしか生成できません。
複数のインスタンスを作成しようとすると例外が発生します。

インポート

```
from PySide6.QtWidgets import QApplication
```

基本的な使用方法

QWidgetについては後述。

以下が最低限のコード。

```
from PySide6.QtWidgets import QApplication, QWidget # 必要なPySide6のクラスをインポート

app = QApplication() # QApplicationインスタンスを作成（アプリケーション全体を管理）
window = QWidget() # 基本的なウィンドウを作成
window.show() # ウィンドウを表示
app.exec() #
```

sys.exit() の引数とすることで、異常終了を検知することができる。

```
from PySide6.QtWidgets import QApplication, QWidget

app = QApplication()
window = QWidget()
window.show()
sys.exit(app.exec()) # イベントループを開始し、アプリケーションの終了コードで終了
```

引数等を受け取って利用することも考えられる。
以下は、コマンドライン引数を受け取って利用する例。

```
import sys # システムモジュールをインポート（コマンドライン引数の処理に必要）
from PySide6.QtWidgets import QApplication, QWidget

app = QApplication(sys.argv) # sys.argv を受け取って利用する
window = QWidget()
window.show()
sys.exit(app.exec())
```

主要なメソッド

コンストラクタ

```
QApplication(argv)
```

パラメータ: | パラメータ | 説明 | |----------|-----| | `argv` | list: コマンドライン引数のリスト（通常は `sys.argv` ） |

戻り値: | 戻り値 | 説明 | |-------|-----| | QApplication | アプリケーションのインスタンス |

例外: | 例外 | 説明 | |-----|-----| | RuntimeError | 既にQApplicationインスタンスが存在する場合 |

exec()

```
app.exec()
```

アプリケーションのメインイベントループを開始します。ユーザーがアプリケーションを終了するまでブロックされます。

パラメータ: | パラメータ | 説明 | |----------|-----| | なし | - |

戻り値: | 戻り値 | 説明 | |-------|-----| | int | アプリケーションの終了コード |

quit()

```
app.quit()
```

アプリケーションを終了します。イベントループを終了し、`exec()`から戻ります。

パラメータ: | パラメータ | 説明 | |----------|-----| | なし | - |

戻り値: | 戻り値 | 説明 | |-------|-----| | None | - |

instance()

```
QApplication.instance()
```

現在のQApplicationインスタンスを取得します。アプリケーション内で1つだけ存在するインスタンスへのアクセスに使用されます。

パラメータ: | パラメータ | 説明 | |----------|-----| | なし | - |

戻り値: | 戻り値 | 説明 | |-------|-----| | QApplication | 現在のアプリケーションインスタンス。インスタンスが存在しない場合はNone |

インスタンス変数とプロパティ

QApplicationクラスは、アプリケーション全体の状態を管理するための様々なプロパティを提供しています。

初心者にとって重要なプロパティ

プロパティ	説明	設定メソッド	取得メソッド
`style`	アプリケーションのスタイル	`setStyle(style)`	`style()`
`font`	アプリケーションのデフォルトフォント	`setFont(font)`	`font()`
`applicationName`	アプリケーションの名前	`setApplicationName(name)`	`applicationName()`
`applicationVersion`	アプリケーションのバージョン	`setApplicationVersion(version)`	`applicationVersion()`

そのほかのプロパティの例

プロパティ	説明	設定メソッド	取得メソッド
`organizationName`	組織名	`setOrganizationName(name)`	`organizationName()`
`organizationDomain`	組織のドメイン名	`setOrganizationDomain(domain)`	`organizationDomain()`
`palette`	アプリケーションのパレット	`setPalette(palette)`	`palette()`
`layoutDirection`	レイアウトの方向	`setLayoutDirection(direction)`	`layoutDirection()`

状態プロパティ（初心者向け）

プロパティ	説明	取得メソッド
`activeWindow`	現在アクティブなウィンドウ	`activeWindow()`
`focusWidget`	現在フォーカスされているウィジェット	`focusWidget()`
`mouseButtons`	現在押されているマウスボタン	`mouseButtons()`
`keyboardModifiers`	現在押されているキーボード修飾子	`keyboardModifiers()`

使用例

```
app = QApplication(sys.argv)

# アプリケーション情報の設定
app.setApplicationName("My App")
app.setApplicationVersion("1.0.0")
```

```
app.setOrganizationName("My Company")
app.setOrganizationDomain("example.com")

# スタイルとフォントの設定
app.setStyle("Fusion")  # モダンなスタイル
app.setFont(QFont("Arial", 10))

# 状態の取得
active_window = app.activeWindow()
focused_widget = app.focusWidget()
```

QWidgetのイベントハンドラ

QWidgetクラスは、様々なイベントを処理するためのハンドラメソッドを提供しています。これらのメソッドは、必要に応じてオーバーライドして使用できます。

イベントハンドラー一覧

カテゴリ	メソッド	説明
ウィンドウ	`closeEvent(self, event)`	ウィンドウを閉じる時
	`showEvent(self, event)`	ウィンドウが表示される時
	`hideEvent(self, event)`	ウィンドウが隠される時
	`moveEvent(self, event)`	ウィンドウが移動される時
	`resizeEvent(self, event)`	ウィンドウのサイズが変更される時
マウス	`mousePressEvent(self, event)`	マウスボタンが押された時
	`mouseReleaseEvent(self, event)`	マウスボタンが離された時
	`mouseMoveEvent(self, event)`	マウスが移動した時
	`mouseDoubleClickEvent(self, event)`	マウスがダブルクリックされた時
	`wheelEvent(self, event)`	マウスホイールが回転した時
キーボード	`keyPressEvent(self, event)`	キーが押された時
	`keyReleaseEvent(self, event)`	キーが離された時
フォーカス	`focusInEvent(self, event)`	ウィジェットがフォーカスを得た時
	`focusOutEvent(self, event)`	ウィジェットがフォーカスを失った時
ドラッグ＆ドロップ	`dragEnterEvent(self, event)`	ドラッグがウィジェットに入った時
	`dragLeaveEvent(self, event)`	ドラッグがウィジェットから出た時
	`dropEvent(self, event)`	ドロップされた時
描画	`paintEvent(self, event)`	ウィジェットの描画が必要な時

イベントハンドラの使用例

ウィンドウを閉じる時の処理

```
class MyWindow(QWidget):
    def closeEvent(self, event):
        print("ウィンドウを閉じます")
        event.accept()  # イベントを処理したことを通知
```

マウスの動きを追跡

```
class MyWindow(QWidget):
    def mouseMoveEvent(self, event):
        print(f"マウス位置: ({event.x()}, {event.y()})")
```

キー入力を処理

```
class MyWindow(QWidget):
    def keyPressEvent(self, event):
        if event.key() == Qt.Key_Escape:
            print("ESCキーが押されました")
            self.close()
```

使用例

基本的なアプリケーション

```
 import sys
from PySide6.QtWidgets import QApplication, QMainWindow

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec())
```

注意事項

- 1つのアプリケーションにつき、QApplicationのインスタンスは1つだけ作成してください
- `sys.exit(app.exec())` でアプリケーションを適切に終了させることが重要です