QMessageBox クラス

概要

`QMessageBox` は、ユーザーに情報を表示したり、重要な質問への回答を求めたりするためのモーダルダイアログボックスです。情報表示、警告、エラー報告、確認ダイアログなど、様々なタイプのメッセージ表示に使用され、デスクトップアプリケーションには欠かせないコンポーネントです。

基本的な使用方法

```
from PySide6.QtWidgets import QApplication, QMessageBox, QPushButton, QWidget
import sys

app = QApplication(sys.argv)

# 基本的なメッセージボックスの表示
QMessageBox.information(None, "タイトル", "これは情報メッセージです。")

app.exec()
```

メッセージボックスの種類

QMessageBoxは以下のタイプのメッセージを提供します：

タイプ	アイコン	用途	静的メソッド
Information	情報アイコン	一般的な情報の表示	`QMessageBox.information()`
Warning	警告アイコン	注意事項や警告	`QMessageBox.warning()`
Critical	エラーアイコン	エラーや重大な問題	`QMessageBox.critical()`
Question	質問アイコン	ユーザーへの質問	`QMessageBox.question()`

静的メソッドによる簡単な使用

情報表示

```
# 基本的な情報表示
QMessageBox.information(parent, "完了", "処理が正常に完了しました。")

# 戻り値を使った例
reply = QMessageBox.information(
    parent,
    "保存完了",
    "ファイルが保存されました。",
    QMessageBox.Ok
)
```

警告表示

```
# 警告メッセージ
QMessageBox.warning(parent, "警告", "この操作は元に戻せません。")

# 複数ボタンでの警告
reply = QMessageBox.warning(
    parent,
    "警告",
    "保存されていない変更があります。続行しますか？",
    QMessageBox.Yes | QMessageBox.No,
    QMessageBox.No  # デフォルトボタン
)

if reply == QMessageBox.Yes:
    print("続行します")
else:
    print("キャンセルしました")
```

エラー表示

```
# エラーメッセージ
QMessageBox.critical(parent, "エラー", "ファイルの読み込みに失敗しました。")

# 詳細なエラー情報
```

```
error_details = "ファイルパス: /path/to/file\nエラーコード: 404"
QMessageBox.critical(
    parent,
    "ファイルエラー",
    f"ファイルが見つかりません。\n\n{error_details}"
)
```

質問ダイアログ

```
# はい/いいえの質問
reply = QMessageBox.question(
    parent,
    "確認",
    "本当に削除しますか？",
    QMessageBox.Yes | QMessageBox.No,
    QMessageBox.No
)

if reply == QMessageBox.Yes:
    # 削除処理
    perform_delete()
```

カスタムメッセージボックス

より細かい制御が必要な場合は、QMessageBoxオブジェクトを直接作成します：

基本的なカスタムボックス

```
# カスタムメッセージボックスの作成
msg_box = QMessageBox()
msg_box.setWindowTitle("カスタム確認")
msg_box.setText("メインメッセージがここに表示されます。")
msg_box.setInformativeText("追加の詳細情報をここに表示できます。")
msg_box.setIcon(QMessageBox.Question)

# カスタムボタンの追加
msg_box.setStandardButtons(QMessageBox.Save | QMessageBox.Discard | QMessageBox.Cancel)
msg_box.setDefaultButton(QMessageBox.Save)

# 実行と結果の取得
result = msg_box.exec()

if result == QMessageBox.Save:
    print("保存が選択されました")
elif result == QMessageBox.Discard:
    print("破棄が選択されました")
else:
    print("キャンセルされました")
```

詳細な情報付きメッセージボックス

```
msg_box = QMessageBox()
msg_box.setWindowTitle("処理結果")
msg_box.setText("データの処理が完了しました。")
msg_box.setInformativeText("一部のファイルで警告が発生しました。詳細を確認しますか？")

# 詳細なテキストを設定
detailed_text = """
処理されたファイル: 150個
成功: 147個
警告: 3個
エラー: 0個

警告があったファイル:
- file1.txt: 文字エンコーディングの問題
- file2.csv: 不正な日付形式
- file3.json: 不明なフィールド
"""
msg_box.setDetailedText(detailed_text)

msg_box.setStandardButtons(QMessageBox.Ok)
msg_box.exec()
```

標準ボタンの種類

ボタン	説明	使用場面
`Ok`	OK	確認・了承
`Cancel`	キャンセル	操作の中止
`Yes`	はい	肯定的な回答
`No`	いいえ	否定的な回答
`Save`	保存	データの保存

ボタン	説明	使用場面
`Discard`	破棄	変更の破棄
`Apply`	適用	設定の適用
`Reset`	リセット	初期状態に戻す
`Close`	閉じる	ダイアログを閉じる
`Help`	ヘルプ	ヘルプの表示

実用的な使用例

1. ファイル保存の確認

```
def confirm_save_file():
    if has_unsaved_changes():
        reply = QMessageBox.question(
            self,
            "未保存の変更",
            "ファイルに未保存の変更があります。\n保存してから続行しますか？",
            QMessageBox.Save | QMessageBox.Discard | QMessageBox.Cancel,
            QMessageBox.Save
        )

        if reply == QMessageBox.Save:
            return save_file()
        elif reply == QMessageBox.Discard:
            return True
        else:  # Cancel
            return False
    return True
```

2. アプリケーション終了の確認

```
def closeEvent(self, event):
    """アプリケーション終了時の確認"""
    reply = QMessageBox.question(
        self,
        "終了確認",
        "アプリケーションを終了しますか？",
        QMessageBox.Yes | QMessageBox.No,
        QMessageBox.No
    )

    if reply == QMessageBox.Yes:
        event.accept()
    else:
        event.ignore()
```

3. 操作結果の報告

```
def show_operation_result(success_count, error_count):
    if error_count == 0:
        QMessageBox.information(
            self,
            "処理完了",
            f"すべての処理が正常に完了しました。\n処理件数: {success_count}件"
        )
    else:
        msg_box = QMessageBox()
        msg_box.setIcon(QMessageBox.Warning)
        msg_box.setWindowTitle("処理完了（警告あり）")
        msg_box.setText("処理が完了しましたが、いくつかのエラーが発生しました。")
        msg_box.setInformativeText(f"成功: {success_count}件\nエラー: {error_count}件")
        msg_box.setStandardButtons(QMessageBox.Ok)
        msg_box.exec()
```

4. 条件付きメッセージ表示

```
def show_conditional_message(user_level, operation):
    if user_level == "beginner":
        QMessageBox.information(
            self,
            "ヒント",
            f"{operation}を実行しました。\n\n"
            "💡 ヒント: この機能は設定画面でカスタマイズできます。"
        )
    elif operation == "delete" and user_level != "expert":
        QMessageBox.warning(
            self,
            "削除実行",
            "アイテムが削除されました。\n\n"
            "⚠️ 注意: 削除されたアイテムは復元できません。"
        )
```

カスタムアイコンとスタイリング

カスタムアイコンの設定

```
from PySide6.QtGui import QIcon, QPixmap

msg_box = QMessageBox()
msg_box.setWindowTitle("カスタムメッセージ")
msg_box.setText("カスタムアイコン付きのメッセージです。")

# カスタムアイコンを設定
custom_icon = QIcon("custom_icon.png")
msg_box.setIconPixmap(custom_icon.pixmap(64, 64))

msg_box.exec()
```

スタイルシートの適用

```
msg_box = QMessageBox()
msg_box.setStyleSheet("""
    QMessageBox {
        background-color: #f0f0f0;
        color: #333333;
    }
    QMessageBox QPushButton {
        background-color: #007acc;
        color: white;
        border: none;
        padding: 8px 16px;
        border-radius: 4px;
        min-width: 80px;
    }
    QMessageBox QPushButton:hover {
        background-color: #005a9e;
    }
""")
```

非モーダルメッセージボックス

通常のメッセージボックスはモーダル（他の操作をブロック）ですが、非モーダルな表示も可能です：

```
def show_non_modal_message():
    msg_box = QMessageBox()
    msg_box.setWindowTitle("非モーダルメッセージ")
    msg_box.setText("この警告は他の操作をブロックしません。")
    msg_box.setIcon(QMessageBox.Information)
    msg_box.setStandardButtons(QMessageBox.Ok)

    # 非モーダルで表示
    msg_box.setModal(False)
    msg_box.show()  # exec()ではなくshow()を使用
```

ベストプラクティス

1. 適切なメッセージタイプの選択

```
# 良い例：適切なタイプを使用
QMessageBox.critical(self, "エラー", "ネットワーク接続に失敗しました。")  # エラー用
QMessageBox.warning(self, "警告", "ディスク容量が不足しています。")      # 警告用
QMessageBox.information(self, "完了", "バックアップが完了しました。")    # 情報用
```

2. 明確で具体的なメッセージ

```
# 良い例：具体的なメッセージ
QMessageBox.question(
    self,
    "ファイル削除の確認",
    f"ファイル '{filename}' を完全に削除しますか？\n\nこの操作は元に戻せません。",
    QMessageBox.Yes | QMessageBox.No,
    QMessageBox.No
)

# 避けるべき例：曖昧なメッセージ
QMessageBox.question(self, "確認", "実行しますか？")
```

3. 適切なデフォルトボタンの設定

```
# 破壊的な操作では安全なオプションをデフォルトに
reply = QMessageBox.warning(
```

```
    self,
    "データ削除",
    "すべてのデータが削除されます。続行しますか？",
    QMessageBox.Yes | QMessageBox.No,
    QMessageBox.No  # 安全なオプションをデフォルトに
)
```

注意事項

1. **ユーザビリティ**: メッセージは簡潔で分かりやすくする
2. **アクセシビリティ**: 適切なタイトルとアイコンを使用する
3. **国際化**: テキストは翻訳可能な形で管理する
4. **パフォーマンス**: 頻繁に表示されるメッセージは適度に制限する

関連するクラス

- **QDialog**: カスタムダイアログの基底クラス
- **QInputDialog**: 入力を求めるダイアログ
- **QFileDialog**: ファイル選択ダイアログ
- **QColorDialog**: 色選択ダイアログ
- **QProgressDialog**: 進行状況表示ダイアログ

参考リンク

- Qt公式ドキュメント - QMessageBox
- PySide6公式ドキュメント