

MetroStyleDeveloper(プチ)

#05 大阪

～メトロスタイルアプリ エキスパート塾++1～

2012/8/30 THU

遥佐保

Microsoft MVP for Client App Dev [Jan,2010 - Dec,2012]

Topics

PPL Taskの書き方

非同期取得を考える

- asynchronous

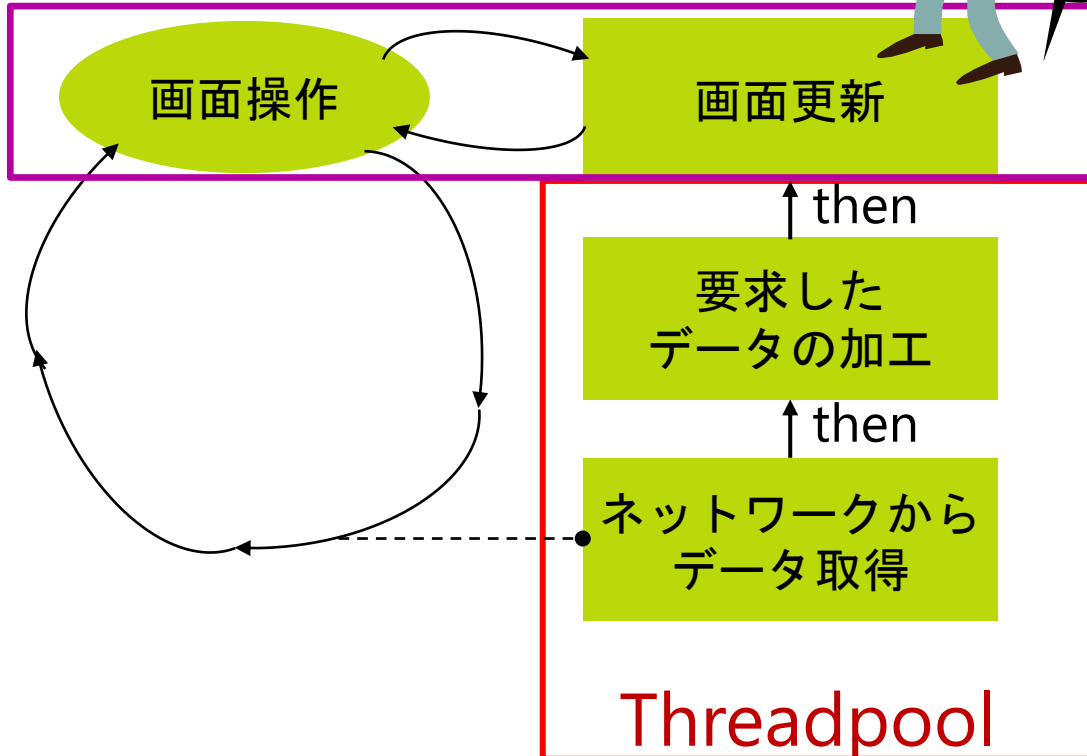
画面(UI)が固まる原因



Web作製指針

- 0.5秒停止で
 - ・ 使いにくい
- 1秒停止で
 - ・ 固まった
- 3秒停止で
 - ・ 壊れてる

データ取得は非同期が流行り



Async

- 細かくたくさんのリクエストを投げる
- 同時にたくさん動かせる (複数CPUの活用！)
- キャンセル操作もしやすい

PPLTaskを利用してみる

PPL Task

```
task<int> t([]()  
{  
    return (1);  
});
```

同じ型

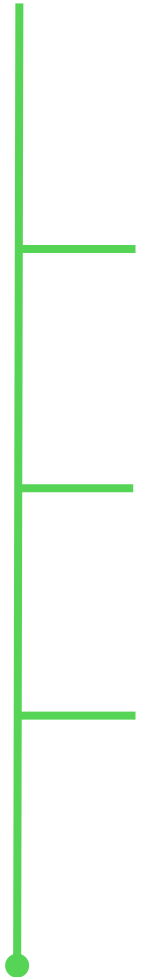
```
task<void> t2 = t.then ([](int n)  
{  
    return( n+1 );  
});
```

.then のreturnは 次のタスク
に引き継がれる

```
task<void> t3 = t.then ([](int n)  
{  
    return( n+1 );  
});  
int result = t.get()
```

つなげて書くと

```
task<int> t([]()  
{  
    return 1;  
}).then([](int n)  
{  
    return n+1;  
}).then([](int n)  
{  
    return n+1;  
}).then([](int n)  
{  
    return n+1;  
})
```



Taskのつなぎ方

```
task<int> tasks[] = {t1, t2, t3};  
auto taskResult = when_all (begin(tasks), end(tasks))  
    .then([](std::vector<int> results) {
```

```
(t1 && t2 && t3).then // 全部終わったあと
```

```
task<int> tasks[] = {t1, t2, t3};  
auto taskResult = when_any (begin(tasks), end(tasks))  
    .then([](int result) {
```

```
(t1 || t2 || t3).then // どれか終わったあと
```

ファイルの読み込みなど
作ってみる

Access Documents-Package.appxmanifest

App1 - Microsoft Visual Studio

ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) SQL(Q) ツール(T) テスト(S) アーキテクチャ

ローカル コンピューター - ネイティブのみ - Debug - Win32

ソリューション エクスプローラー

- ソリューション 'App1' (1 プロジェクト)
- App1
 - 共通
 - 外部依存関係
 - 資産
 - Logo.png
 - SmallLogo.png
 - SplashScreen.png
 - StoreLogo.png
 - App.xaml
 - App.xaml.cpp
 - App.xaml.h
 - App1_TemporaryKey.pfx
 - MainPage.xaml
 - MainPage.xaml.cpp
 - MainPage.xaml.h
 - Package.appxmanifest
 - pch.cpp
 - pch.h

出力 Package.appxmanifest* MainPage.xaml.h

アプリケーションの配置パッケージのプロパティはアプリケーション マニフェスト ファイルに格納されま

アプリケーション UI 機能 宣言 パッケージ化

このページを使用して、アプリケーションで使用できるシステムの機能またはデバイスを指定します

機能:

- ドキュメント ライブラリ
- エンタープライズ認証
- インターネット (クライアント)
- インターネット (クライアントとサーバー)
- 場所
- マイク
- 音楽ライブラリ
- 画像ライブラリ
- プライベート ネットワーク (クライアントとサーバー)
- 近接
- リムーバブル記憶域
- 共有ユーザー証明書
- ビデオ ライブラリ
- Web カメラ

説明:

この機能は、ストア ポリシーの対象のドキュメント ライブラリ内のファイル 連付け宣言で定義された、ドキュメ ター上のドキュメント ライブラリにはア

[詳細](#)

アプリケーション UI 機能 宣言 パッケージ化

このページを使用して、宣言の追加および宣言のプロパティの指定を行います。

使用可能な宣言:

ファイルの種類に関連付け

サポートされる宣言:

ファイルの種類に関連付け

説明:

アプリケーションの代わりに、ファイルの種類を各アプリケーションで、この宣言の複数のインスタンス

[詳細](#)

プロパティ:

表示名:

ロゴ:

ヒント:

名前:

編集フラグ

- 安全なファイルを開く
- 常に安全でない

サポートされるファイルの種類

少なくとも 1 つのファイルの種類がサポートされ ("jpg")。

サポートされるファイルの種類

コンテンツ タイプ:

ファイルの種類

MainPage.xaml



MainPage.xaml.h

private:

```
Windows::Storage::StorageFile^ my_file;
```

(以下は自動で作ってくれます)

```
void my_btn_01_click(Platform::Object^ sender,  
                    Windows::UI::Xaml::RoutedEventArgs^ e);
```

```
void my_btn_02_click(Platform::Object^ sender,  
                    Windows::UI::Xaml::RoutedEventArgs^ e);
```

```
void my_btn_03_click(Platform::Object^ sender,  
                    Windows::UI::Xaml::RoutedEventArgs^ e);
```

MainPage.xaml.cpp(0)

```
// Task利用に必要なインクルードファイル  
#include <ppltasks.h>
```

```
// ハンズオンで利用する名前空間たち
```

```
using namespace concurrency;  
using namespace Windows::Media::Capture;  
using namespace Windows::Foundation;  
using namespace Windows::UI::Xaml::Media::Imaging;
```

MainPage.xaml.cpp(1)

```
void App1::MainPage::my_btn_01_click(
Platform::Object^ sender, Windows::UI::Xaml::RoutedEventArgs^ e)
{
    task<StorageFile^>(
        KnownFolders::DocumentsLibrary->CreateFileAsync(
            my_txt->Text,
            CreationCollisionOption::ReplaceExisting))
        .then([this](StorageFile^ file)
            {
                my_btn_01->Content = "作られたよ";
                my_file = file;
            });
}
```

MainPage.xaml.cpp(2)

```
void App1::MainPage::my_btn_02_click(
Platform::Object^ sender, Windows::UI::Xaml::RoutedEventArgs^ e)
{
    StorageFile^ file = my_file;
    if (file != nullptr){
        String^ wtn_str = my_txt->Text;
        if ((wtn_str != nullptr) && (!wtn_str->IsEmpty())){
            create_task(FileIO::WriteTextAsync(file, wtn_str))
                .then([this, file, wtn_str](task<void> task)
                {
                    try{
                        task.get();
                        my_txt->Text = "書き込んだよ";
                    }catch(COMException^ ex){
                        my_txt->Text = "書き込み失敗したよ0";
                    }
                });
        }else{
            my_txt->Text = "書き込み失敗したよ1";
        }
    }
}
```


MainPage.xaml.cpp(3)

```
void App1::MainPage::my_btn_02_click(
Platform::Object^ sender, Windows::UI::Xaml::RoutedEventArgs^ e)
{
    StorageFile^ file = my_file;
    if (file != nullptr){
        create_task(FileIO::ReadTextAsync(file)).then([this, file](task<String^> task)
        {
            try
            {
                String^ fileContent = task.get();
                my_txt->Text = "ファイル読んだよ '" + file->Name + "':¥n¥n" + fileContent;
            }
            catch(COMException^ ex)
            {
                my_txt->Text = "読み込み失敗したよ";
            }
        });
    }
}
```

カメラ撮影など
作ってみる

MainPage.xaml

```
<Button x:Name="ResetButton" Click="ResetButton_Click_1"  
    Content="ResetButton" HorizontalAlignment="Left"  
    VerticalAlignment="Top" />
```

```
<Image x:Name="CapturedPhoto"  
    AutomationProperties.Name="A placeholder image" Stretch="Fill"  
    HorizontalAlignment="Left" VerticalAlignment="Top"  
    Height="216" Width="384" />
```

MainPage.xaml.cpp(4)

```
}).then([this](task<void> task)
{
    try{
        CameraCaptureUI^ dialog = ref new CameraCaptureUI();
        dialog->PhotoSettings->CroppedAspectRatio = Size(16, 9);
        concurrency::task<StorageFile^> (dialog->CaptureFileAsync(CameraCaptureUIMode::Photo))
            .then([this] (StorageFile^ file){
                if (nullptr != file){
                    concurrency::task<Streams::IRandomAccessStream^> (file->OpenAsync(FileAccessMode::Read)).then([this]
                    (Streams::IRandomAccessStream^ stream){
                        BitmapImage^ bitmapImage = ref new BitmapImage();
                        bitmapImage->SetSource(stream);
                        CapturedPhoto->Source = bitmapImage;
                        ResetButton->Visibility = Windows::UI::Xaml::Visibility::Visible;
                    });
                }else{
                    my_txt->Text = "カメラ撮影失敗したよ0";
                }
            });
    }catch (Platform::Exception^ ex){
        my_txt->Text = "カメラ撮影失敗したよ0";
    }
});
```

Stringのお話は次回に...

- .NETの場合

- System::String

- C++/CXの場合

- Platform::String

- char16*

- <http://msdn.microsoft.com/ja-jp/library/hh755812%28v=vs.110%29.aspx>