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

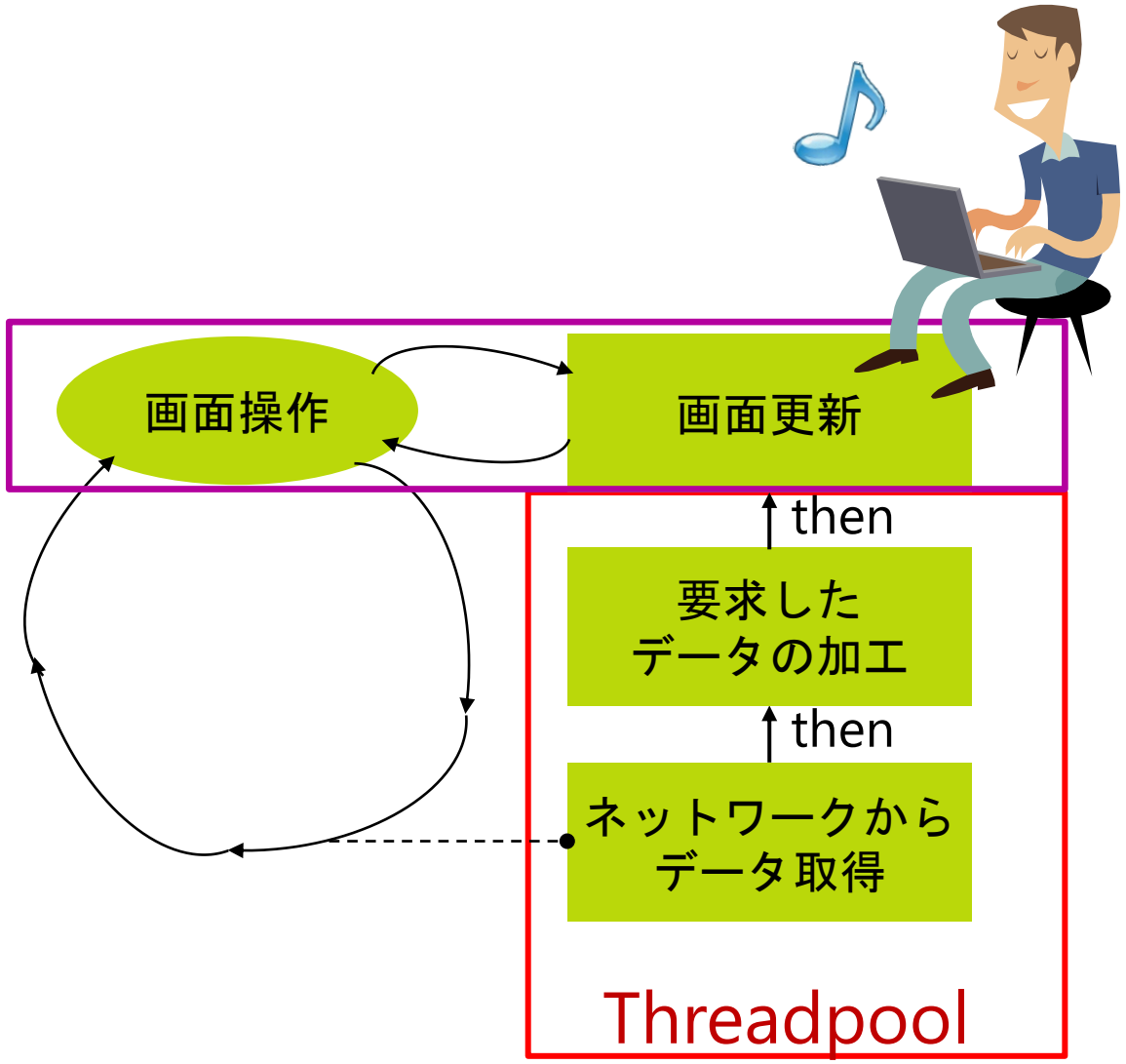
非同期取得を考える
- asynchronous

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

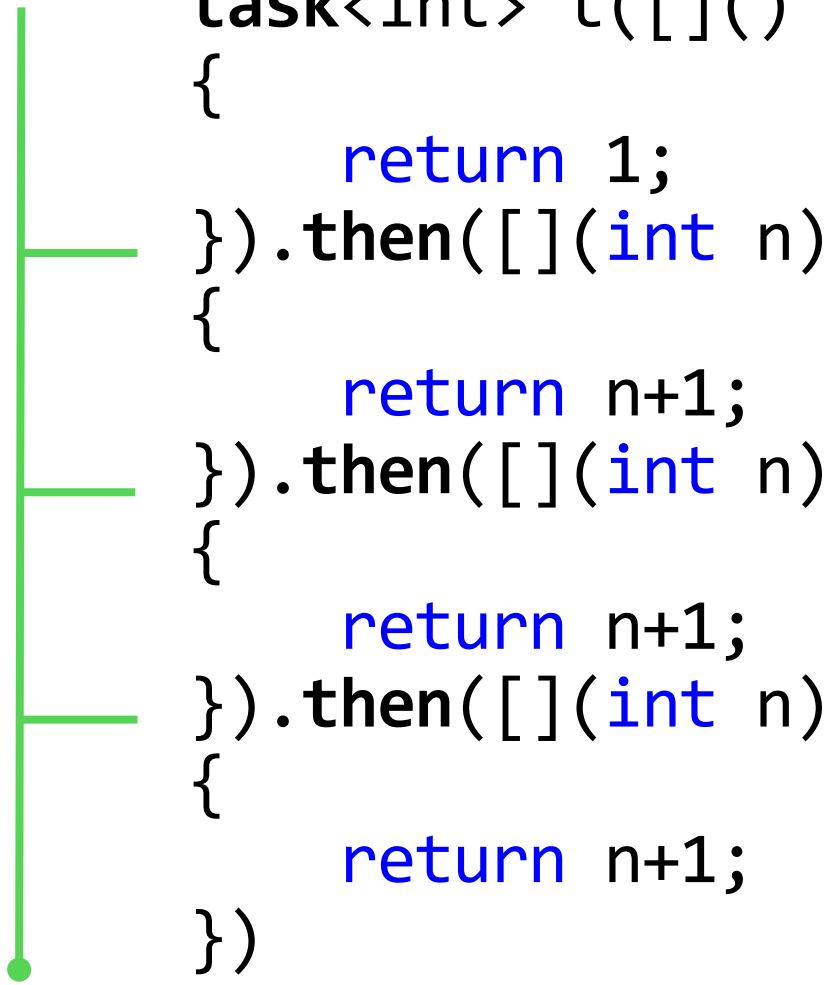
PPL Task

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

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

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

同じ型

.then のreturnは 次のタスクに引き継がれる

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;
})
```

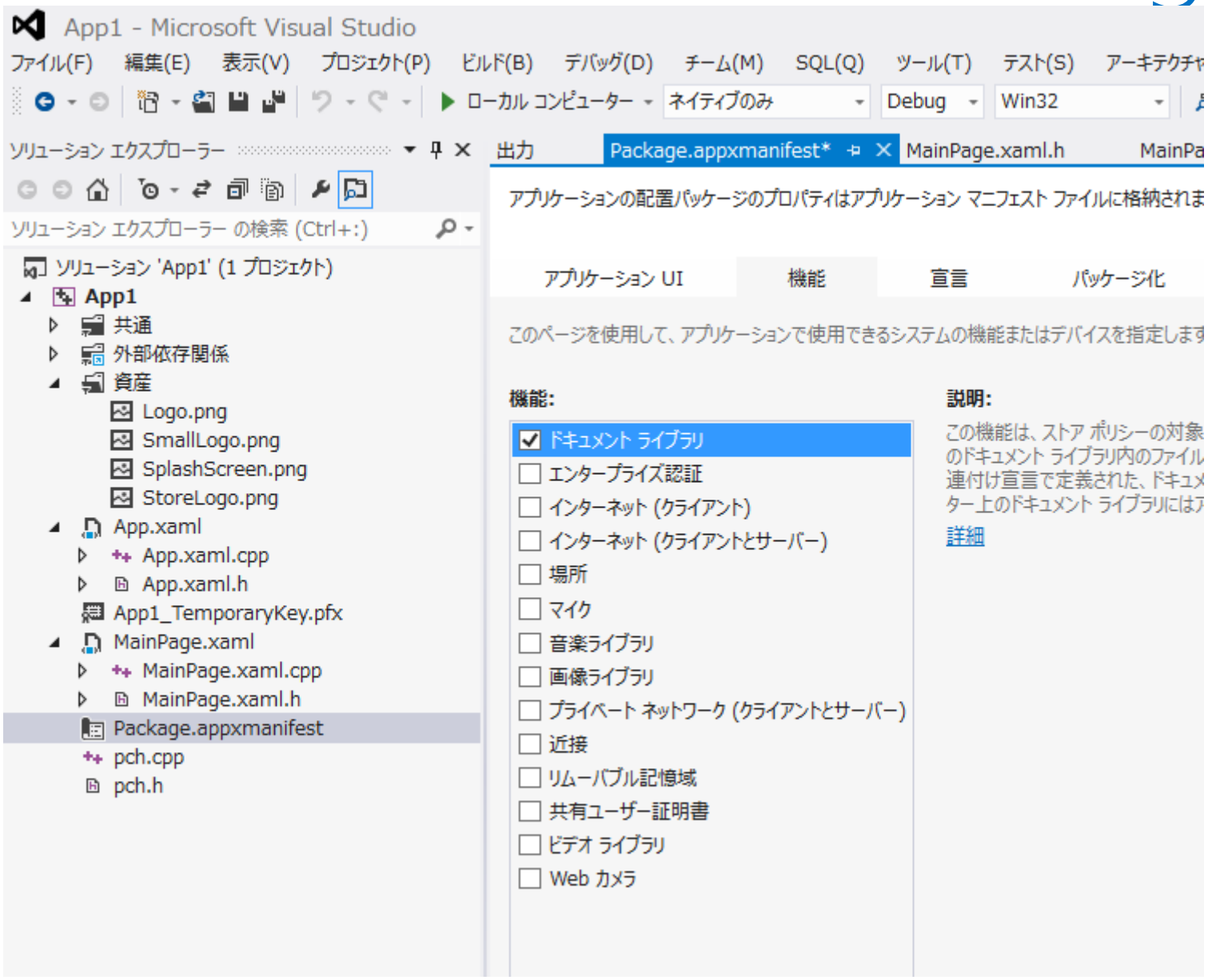
Taskのつなぎ方

```
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-Pakage.appxmanifest



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