

Async 编程简介

ASYNC是什么？



- Async是下一代异步编程模型，Windows RT中包含了一些Asynchronous APIs来确保你的应用在等待的过程中能保持响应性的工作。
- 从代码上看，可以理解为异步同步化，用写同步代码的方式完成异步的工作，省去了以往处理异步事件回调函数的繁琐工作。
- 一般来说，异步的方法在命名时都会以“Async”结尾。

await & async 关键字

Async模式的实现需要用到两个关键字，`async`，`await`。

用`async`修饰一个方法为异步方法，如果没有`await`配对的关键字，那么方法默认以顺序的方式执行。如果有`await`，那么会在`await`处挂起，并将控制权返回给调用者以执行其他的操作。一般`async`修饰的方法的返回值是`void`，`Task`，或`Task<TResult>`。

```
private async void button_Click(object sender, EventArgs e)
{
    (sender as Button).Enabled = false;

    StorageFolder local = ApplicationData.Current.LocalFolder;
    StorageFile file = await local.CreateFileAsync("filename");

    (sender as Button).Enabled = true;
}
```

Task 类

Task类提供了一些Run方法，可以做一些耗时的工作，类似开辟一个新的线程，并能提供返回值。

可以通过Task类来创建自定义异步方法。

```
// 创建自定义异步方法
public async Task<string> GetString()
{
    return Task<string>.Run(()=>
    {
        // ... do something
        return "I am a string";
    });
}

// 调用自定义异步方法
public async void CallGetString()
{
    string str = await GetString();
    // ... do something
}
```

Task类的更多特性

连续性

```
Task task = Task.Run(() =>
{
    
```

```
// ... do something
return 5;
}).ContinueWith(prevTask =>
{
    int prevResult = prevTask.Result;
    // ... do something
    return prevResult * 2;
}).ContinueWith(prevTask2 =>
{
    // ... do something
}, TaskScheduler.FromCurrentSynchronizationContext());
```

并行性

```
Task task1 = Task.Run(() => { });
Task task2 = Task.Run(() => { });

Task.WaitAll(task1, task2); // 等待提供的所有 System.Threading.Tasks.Task 对象完成执行过程。
Task.WaitAny(task1, task2); // 等待提供的任何 System.Threading.Tasks.Task 对象完成执行过程。
```