

Threading and Services



Outline

- Asynchronous operations and threading
- Data binding and services
- WPF and WCF

Responsive UI & Distributed Systems

- **Distributed systems are slow**

- Network congestion
- Busy servers
- Packet loss
- Round trip times

- **Local IO can be slow too**

- **Basic design rule:**

- Don't freeze the UI

WPF Threading Model

- **Window (and all its content) has a Dispatcher**
 - All input delivered via Dispatcher
 - Equivalent of message pump
- **Dispatcher belongs to a thread**
 - Elements must be accessed from their Dispatcher's thread
 - Same 'STA' model as Win32 and COM

```
using System.Windows.Threading;

class DispatcherObject
{
    public Dispatcher Dispatcher { get; }

    public bool CheckAccess();
    public void VerifyAccess();
}
```

The Golden Rule of Threading

- Same as Windows Forms:

**Don't touch the UI from
the wrong thread!**

- Easier than Windows Forms
 - WPF tells you when you get it wrong!

The Golden Rule of Responsiveness

- Dispatcher thread handles input
- Corollary:

Don't block the UI thread!

Resolving the Rules

**Don't touch the UI from
the wrong thread!**

Don't block the UI thread!

- So how do I get any work done?

Async Work

- Perform slow work asynchronously
- Update UI using `Dispatcher.BeginInvoke`

```
ThreadStart ts = delegate // Can use any delegate type
{
    DoSlowWork();

    Dispatcher.BeginInvoke(DispatcherPriority.Normal, (EventHandler) delegate
    {
        DoUIUpdate();
    }, null, null);
};

ts.BeginInvoke(delegate(IAsyncResult iar) { ts.EndInvoke(iar); }, null);
```


DispatcherPriority

SystemIdle

ApplicationIdle

ContextIdle

Background

Input

Loaded

Render

DataBind

Normal

Send

DispatcherOperation

- Returned by Dispatcher.BeginInvoke

Operations

Wait

Abort

Properties

Priority

Status

Result

Events

Completed

Aborted

SynchronizationContext

- Not WPF-specific
 - (Also works on Windows Forms)

```
SynchronizationContext syncContext = SynchronizationContext.Current;  
ThreadStart ts = delegate  
{  
    DoSlowWork();  
  
    syncContext.Post(delegate  
    {  
        DoUIUpdate();  
    }, null);  
};  
  
ts.BeginInvoke(delegate(IAsyncResult iar) { ts.EndInvoke(iar); }, null);
```

Asynchronous Options

- .NET v1 asynchronous pattern
- .NET v2.0 event-based asynchronous pattern
 - Aka “Asynchronous Pattern for Components”
- Thread pool
- Create your own thread

.NET v1 Asynchronous Pattern

- (Still very much alive in .NET v2.0)
- **BeginXxx/EndXxx and IAsyncResult**
 - Async delegate invocation
 - Sockets, streams etc.
 - Web service proxies
- **Pros:**
 - Efficient for intrinsically async work
- **Cons:**
 - Outstanding operations unbounded
 - More complexity than most UIs need

.NET 2 Event-Based Async Pattern

- **XxxAsync and CancelAsync/XxxAsyncCancel methods**
- **XxxCompleted and [Xxx]ProgressChanged event**
 - Raised on original caller's thread
- **Pros:**
 - Simpler than v1 async pattern
 - Supports cancellation
 - Presents a *single threaded* face
- **Cons:**
 - Less widely supported
 - Loss of WaitHandle

Thread Pool

- **Implicit use**
 - Async delegate invocation
 - Some async pattern implementations
- **Direct use of thread pool**

Create Your Own Thread

- **Can reduce concurrency:**
 - 2 threads: 1 worker, 1 UI
- **Concurrency is not the goal**
 - Responsiveness
 - Simplicity
- **Roll your own message passing**

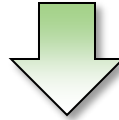
Data Binding and Threading

- **Change notification == UI update**
- **WPF offers some leeway**
 - Property change events allowed on any thread
 - (but not list change events)
 - `ObjectDataProvider.IsAsynchronous`
 - `Binding.IsAsync`
- **Usually better to keep data model on UI thread**

WCF and v1 Async Pattern

svcutil.exe

<http://services.msdn.microsoft.com/ContentServices/ContentService.asmx?wsdl>
/language:C# /async



```
[ServiceContract(Namespace="urn:msdn-com:public-content-syndication",  
                  ConfigurationName="ContentServicePortType")]  
public interface ContentServicePortType  
{  
    ...  
    [OperationContract(AsyncPattern=true,  
        Action="urn:msdn-com:public-content-syndication/GetContent",  
        ReplyAction="*")]  
    IAsyncResult BeginGetContent(GetContentRequest request,  
        AsyncCallback callback, object asyncState);  
  
    GetContentResponse EndGetContent(IAsyncResult result);  
}
```

WCF and Data Binding

- **Generated types:**
 - Provide properties
 - Implement change notifications
- **Usually suitable as data sources**

Summary

- Dispatcher
- Asynchronous options
- SynchronizationContext