

cWebAppBusinessProcess

Peter Bragg



“I’m migrating my windows application to the web”



“I’m migrating my BUSINESS application to the web”



Our old friend the batch process

Browser Timeouts

- › In web applications, a timeout occurs when the client (usually a web browser) makes a request to the server but does not receive a response within a specified time limit.
- › Timeouts help prevent the browser from waiting indefinitely and improve user experience by handling slow or unresponsive servers gracefully.
- › Default is 90 seconds with IIS.



Possible Solutions ...

Solution?

1. We cross our fingers and hope that our process takes less than 90 seconds.
 - › 90 seconds is a long time to wait.
2. We up the timeout limit to something greater than 90 seconds.
 - › Does this make for a good user experience?
 - › How big is big enough?
 - › As said. 90 seconds is already a long time to wait.

Solution?

3. OnReleaseProcess


- › Browser gets a timely response, but
- › Process is locked out of the pool and unavailable for handling the next request.

4. Start a background (non-web) process

- › Runprogram Shell Background
- › Writes progress to database (or other)
- › cWebTimer in WebApp monitors progress and reports back to the UI (end-user)

Solution?

5. Use cWebBusinessProcess? Oh. Obsolete. Ho-hum.



```
// obsolete class. Use cWebComponent or cWebAspClassicObject. See comments at the
// top of this file.
// 19.1: This class is marked private as is cInternetSessionBusinessProcess and
// cRemoteEntryProcess. These classes are only used if you are using old pre-framework
// style web applications. The newer Web Framework was introduced in 17.1. We marked this
// private to remove a considerable amount documentation, which only added confusion when
// using the current framework. This obsolete class will work as before.

{ ClassLibrary=WebApp }
{ DDOHost=True }
{ HelpTopic=cWebBusinessProcess }
{ Visibility=Private }
{ Obsolete=True }
Class cWebBusinessProcess is a cInternetSessionBusinessProcess

    Procedure Construct_object
        Forward Send construct_object

        // this adds an extra layer of safety with process pooling. It is disabled
        // to get maximum speed and because it is usually not needed. Prior to
        // vdf10, this property did not exist and the clear always happened.
        { Category=Data }
        Property Boolean pbClearDDOsOnDetach False
```



Could we create our own?



cWebAppBusinessProcess

Agenda

- › Basic Requirements.
- › Challenges.
- › Procedure Callback.
- › Examples.
- › When and when not to use.
- › What next ...?

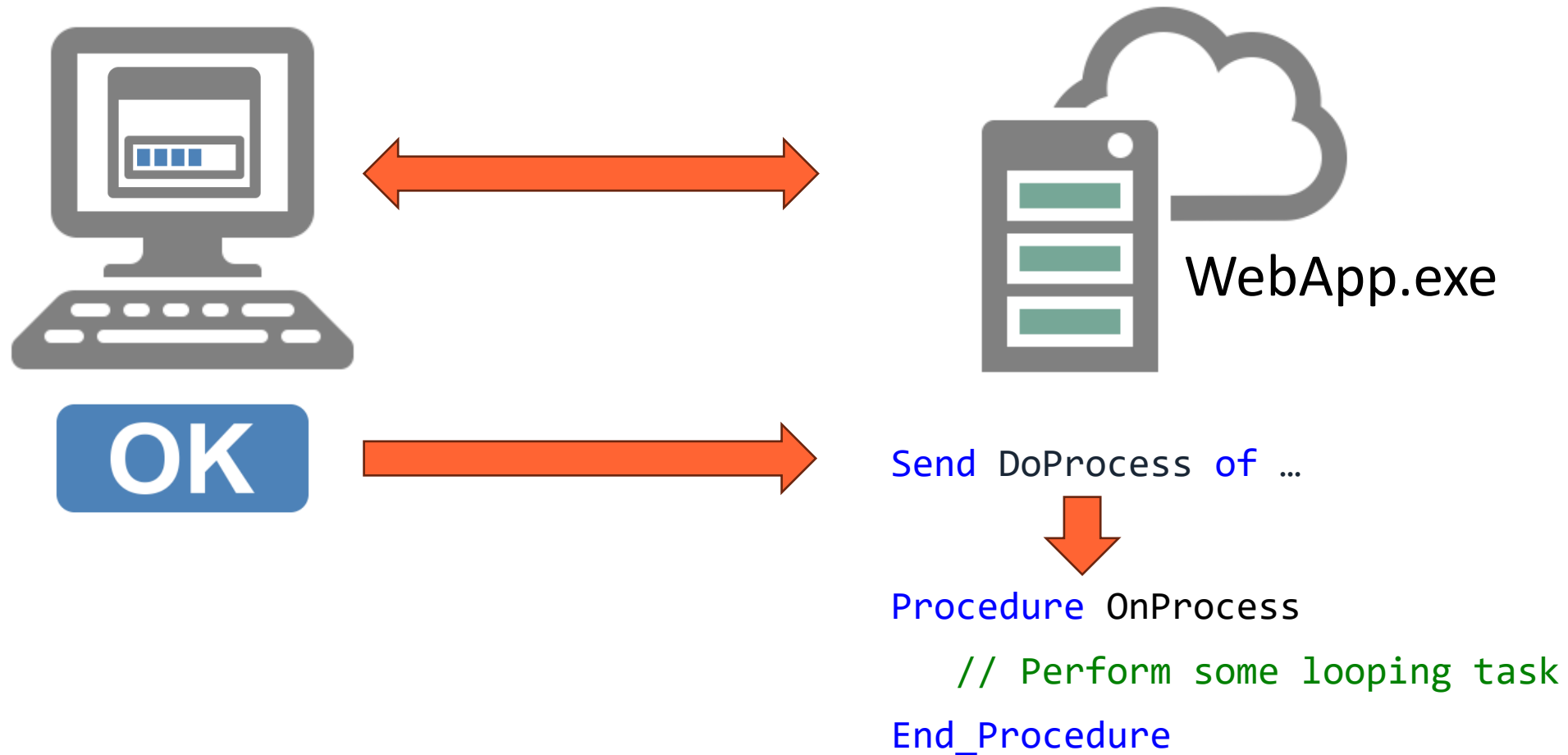


Basic Requirements

cWebAppBusinessProcess Requirements

- › Status Panel.
 - › Browser receives update responses. Timeouts avoided
 - › End user is kept informed of progress
- › Familiar Interface.
 - › Send DoProcess
 - › Code OnProcess
- › Built-in Error handling.
 - › Procedure OnError
 - › Get Error_Count ...

cWebAppBusinessProcess Requirements



cWebAppBusinessProcess Requirements

- › DoProcess is called. Process is started.
- › OnProcess deals with first block of records.
- › We callback to the client with update on progress.
- › Client is updated (timeout avoided). Back to the server.
- › OnProcess deals with next block of records.
- › We callback to the client with update of progress.
- › Client is updated (timeout avoided). Back to the server.
- › Until done.



Process Pooling

cWebAppBusinessProcess Challenges

- › Process Pooling.
- › And it was all going so well. (Well, up to and including the first update of our status panel).
- › For this to work we're going to need a synchronisation process/framework.
- › If only there was something out-the-box that could help?



Procedure Callback

Procedure Callback

Description

Use this procedure to delay a task to a new server next round-trip. This has the advantage that the user interface (UI) will be updated and it will potentially get around timeout issues. You would use this when performing longer running processes.

The message handle passed as the first argument will be called in a new round-trip. This message does need to be published using WebPublishProcedure and should be available on this web object. Up to 10 optional parameters can be passed that will be passed back to the callback message.

```
Object oProgressBar is a cWebProgressBar
    Set piColumnSpan to 7
End_Object

Object oProcessCustomer is a cWebButton
    Set piColumnSpan to 3
    Set psCaption to "Process"
    Set piColumnIndex to 7

    Procedure DoProcessCustomers Integer iProgress
        // Here you would do the actual work
        Sleep 2

        Increment iProgress
        WebSet piValue of oProgressBar to (iProgress * 10)
        If (iProgress < 10) Begin
            Send Callback (RefProc(DoProcessCustomers)) iProgress
        End
    End_Procedure

WebPublishProcedure DoProcessCustomers

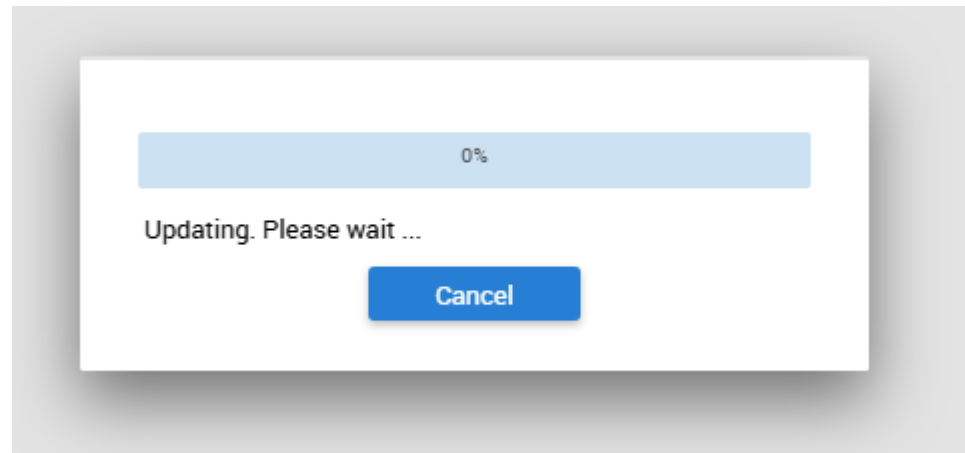
Procedure OnClick
    Send Callback (RefProc(DoProcessCustomers)) 0
End_Procedure
End_Object
```

◀ OnProcess?

◀ DoProcess?

cWebAppBusinessProcess

- › Status Panel.
 - › Make our class a subclass of cWebModalDialog.
 - › Add cWebLabel, cWebProgressBar and cWebButton.



cWebAppBusinessProcess

› Familiar Interface – Properties

```
{ Category = "Behavior" }
```

```
Property Boolean pbAllowCancel      True
```

```
{ Category = "Behavior" }
```

```
Property Integer piFeedbackFrequency 1      // How many OnProcess calls each trip
```

```
{ Category = "Behavior" }
```

```
Property Boolean pbProgressBar      True
```

cWebAppBusinessProcess

› Built-in Error Handling

```
Procedure OnError Integer iErrNum Integer iErrLine String sErrMsg  
End_Procedure
```

```
Function Error_Count Returns Integer  
End_Function
```




Examples

cWebAppBusinessProcess

› Miscellaneous

```
Procedure End_Process  
    Send Ok      // Calls NotifyCloseModalDialog (OnCloseModalDialog)  
End_Procedure
```

```
Procedure OnCloseModalDialog Handle hoWebAppBusinessProcess  
    Integer iErrors  
    Get Error_Count of hoWebAppBusinessProcess to iErrors  
    // Report as required  
End_Procedure
```

cWebAppBusinessProcess Usage

- › Remember – we’re trying to avoid timeouts.
- › Play with piFeedbackFrequency to strike the balance between performance and “waiting”.
- › Avoid “big” SQL updates that cannot provide feedback.
- › Best used when:
 - › Looping through records.
 - › Reading in files.
 - › Performing lots of small admin tasks, e.g. deleting cached files.
 - › IOW lots of doing the same little thing(s).

What Next?

- › Erm. No plans, really.
- › Exercise in understanding Procedure Callback.
- › I will update class and examples on Forum (and make available with presentation slides).
- › By all means someone pick it up and run with it.
- › But ...
- › Data Access? Given we already have Procedure Callback ...

Thank you!

Are there any questions?