BackgroundWorker (Clase)

(Propiedades)

P1 – **CancelationPending** (bool - R)
P2 – **IsBusy** (bool - R)
P3 – **WorkerReportProgress** (bool - R/W)
P4 – **WorkerSupportCancellation** (bool - R/W)

(Metodos)

M1 – **CancelAsync** ()
M2 – **ReportProgress** (int ProgressPrecentage [, object UserState])
M3 – **RunWorkerAsync**([object Argument])

(Handlers de Eventos)

E1 – **DoWork** (object sender, DoWorkEventArgs e) => e.Result (W), e.Argument (R), e.Cancel (W)
E2 – **ProgressChanged** (object sender, ProgressChangedEventArgs e) => e.ProgressPercentage (R), e.UserState (R)
E3 – **RunWorkerCompleted** (object sender, RunWorkerCompletedEventArgs e) => e.Result (R), e.Cancelled (R), e.Error (R)

NOTACION:

[] => argumento o parámetro opcional (no obligatorio)
R, R/W, W => read-only(get), read-write (get/set), write (set)
Red => se usa o ejecuta en un hilo secundario (hilo hijo dentro de **DoWork**)
Blue => se usa o ejecuta en el hilo padre o hilo principal

Disparadores de eventos y cambiadores de Propiedades:

RunWorkerAsync => **DoWork**
ReportProgress => **ProgressChanged**
CancelAsync => **CancelationPending**=true (e.Cancel = true; return;) + e.Cancelled = true
DoWork => **IsBusy** (true = comienza método, false = acaba método) + e.Error (excepción no controlada)

bool	=>	e.Cancel, e.Cancelled	object	=>	e.Result, e.Argument, e.UserState
int	=>	e.ProgressPercentage	Exception	=>	e.Error

Pasos a seguir:

1.- Poner un nombre al control **BackgroundWorker,** acorde al trabajo que va a llevar a cabo en el hilo secundario. Si se va a reportar el progreso del proceso con **ProgressChanged** (*ProgressPercentage*), o a devolver el estado intermedio del objeto *Argument* o similar (*UserState*), marca en **Propiedades** como **true** **WorkerReportProgress**. Si el hilo se va a poder cancelar a voluntad con **CancelAsync()**, marca en **Propiedades** como **true** a **WorkerSupportCancellation**.

2.- Escribimos en el código del UI donde comienza el hilo asíncrono, ejecutando **RunWorkerAsync()**. Así mismo, si vamos a poder cancelar el hilo, escribimos en el código del UI donde cancelarlo con **CancelAsync()**.

3.- Escribimos el código del hilo en **DoWork**. Recogemos el objeto sender (as *BackgroundWorker*) para poder usarlo dentro del código del hilo y acceder a sus propiedades y métodos (encapsulación aislada del hilo secundario del hilo principal). Recogemos con casting en el objeto e.*Argument* pasado desde el UI, mediante **RunWorkerAsync()**. Revisamos si *CancellationPending* es **true**, para entonces poner e.*Cancel* a **true** y salirnos del hilo inmediatamente (**return**). Ejecutamos la tarea del hilo, y podemos reportar su progreso y cualquier cosa al hilo principal mediante el objeto e.*UserState*, si así lo consideramos mediante **ReportProgress()**. Si la tarea del hilo completo ha concluido (no bucle infinito), entonces antes de salir, podemos recoger el resultado en el objeto e.*Result*, mediante polimorfismo (aquí no hace falta el casting por tanto). **IMPORTANTE: Desde este código NO se puede acceder a componentes del UI**.

4.- Si vamos a usar el reporte de progreso, escribimos en el UI el código del método **ProgressChanged()**. Dentro del mismo, podemos leer el contenido del entero e. *ProgressPercentage,* y recuperar mediante casting el valor pasado por el object e.*UserState*. **Desde este código SI se puede acceder a componentes del UI**.

5.- Finalmente escribimos el código en el UI de cuando el hilo acaba normalmente, es cancelado, o salta una excepción incontrolada dentro del mismo. Esto lo hacemos dentro del método handler **RunWorkerCompleted()**. Primero de todo, revisamos si e.*Error* es diferente a **null**, lo que significaría que el hilo se ha salido debido a una excepción no controlada dentro del código del mismo en **DoWork()**. Después revisamos si e.*Cancelled* es **true**, para saber si el hilo ha sido cancelado. Finalmente si nada de esto ha ocurrido, ya podemos, si queremos, acceder al valor devuelto por el hilo mediante casting del objeto e.*Result* (**NUNCA** intentes acceder a él antes de todas estas comprobaciones, o podrá saltar una excepción porque dicho objeto no esta referenciado debido a una salida por error o por cancelación anticipada). **Desde este código SI se puede acceder a componentes del UI**.