On Backround Worker

m\_controller.SessionStartCompleted += OnStartDone;

m\_controller.SessionEndCompleted += OnEndDone;

m\_controller.SessionInfoCompleted += OnInfoDone;

m\_controller.SessionOperatorListCompleted += OnOperatorListDone;

OnStartDone is triggered.

OnInfoDone is triggered.

Solution:

Sol.A

SessionViewModel

private void UpdateSesionListUi()

{

Application.Current.Dispatcher.Invoke(new Action(() =>

{

SessionList.Clear();

foreach (var session in m\_controller.Sessions.OrderByDescending(s => s.Number))

{

SessionList.Add(session);

}

ReportsViewModel.Instance.UpdateReportUIOnStartNewSession();

}));

}

----------------------------------------------------------------------------

**Whats really happening on the background worker>?**

SessionStartCommand = new RelayCommand(parameter => StartSession());

private void StartSession()

{

SessionStatusMessage = Resources.SessionStartProgress;

//if multioperator, then send selected operator

//else send default operator

if (Settings.IsMultiOperator)

{

if (SelectedOperator != null)

{

m\_controller.SessionStart(SelectedOperator);

}

}

else

{

m\_controller.SessionStart(new Business.Operator(0, "B3 Session"));

}

}

public void SessionStart(Operator op)

{

if (op == null)

throw new ArgumentNullException();

if (!IsBusy)

{

IsBusy = true;

ArrayList args = new ArrayList {op.OperatorId, op.OperatorName};

RunWorker(Resources.SessionStartProgress,

DoSessionStart,

args,

false,

OnSessionStartCompleted);

}

}

protected void RunWorker(string progressText, DoWorkEventHandler doWorkHandler, object argument, bool reportProgress, RunWorkerCompletedEventHandler completeHandler)

{

// Set the progress.

ProgressText = progressText;

// Create the worker thread and run it.

BackgroundWorker worker = new BackgroundWorker();

worker.WorkerReportsProgress = reportProgress;

worker.WorkerSupportsCancellation = false;

worker.DoWork += doWorkHandler;

if(reportProgress)

worker.ProgressChanged += OnProgressChanged;

worker.RunWorkerCompleted += completeHandler;

if(argument != null)

worker.RunWorkerAsync(argument);

else

worker.RunWorkerAsync();

}

private void DoSessionStart(object sender, DoWorkEventArgs e)

{

EliteModule.SetThreadCulture(Parent.Settings);

ArrayList args = (ArrayList)e.Argument;

SessionStartMessage message = new SessionStartMessage((int)args[0], (string)args[1]);

message.Send();

if (message.ReturnCode == ServerReturnCode.Success)

{

//DoSessionListNow();

}

else

{

throw new B3CenterException(string.Format(CultureInfo.CurrentCulture, Resources.SessionStartFailed,

ServerErrorTranslator.GetReturnCodeMessage(message.ReturnCode)));

}

}

private void OnSessionStartCompleted(object sender, RunWorkerCompletedEventArgs e)

{

IsBusy = false;

ProgressText = null;

if (!CheckForError(e.Error))

{

}

AsyncCompletedEventHandler handler = SessionStartCompleted;

if (handler != null)

handler(this, new AsyncCompletedEventArgs(e.Error, e.Cancelled, null));

}

public event AsyncCompletedEventHandler SessionStartCompleted;

public void Initialize(B3Controller controller)

{

if (controller == null)

throw new ArgumentNullException();

m\_controller = controller;

PropertyChangedEventManager.AddListener(m\_controller, this, string.Empty); // Listen for changes to the parent and children.

m\_controller.SessionStartCompleted += OnStartDone;

m\_controller.SessionEndCompleted += OnEndDone;

m\_controller.SessionInfoCompleted += OnInfoDone;

m\_controller.SessionOperatorListCompleted += OnOperatorListDone;

m\_controller.GetSessionList();

SelectedBalls = new List<int>(m\_controller.GameBallList);

DisableB3Features();

EnableB3Features(controller.ModuleFeatureList);

if (m\_controller.Session != null && m\_controller.Session.Active)

{

SessionStatusMessage = Resources.SessionStarted;

}

else

{

SessionStatusMessage = Resources.SessionEnded;

}

}

private void OnStartDone(object sender, AsyncCompletedEventArgs e)

{

if (e.Error == null)

{

SetIconStatus(true);

SessionStatusMessage = Resources.SessionStartSuccess;

//After starting a new session, the new session does not show in the report parameter field "Session" .

}

else

{

SetIconStatus(false);

SessionStatusMessage = Resources.SessionStartFailed;

DisplayMessageBox(string.Format(Resources.SessionStartFailed, e.Error));

}

}

private void OnInfoDone(object sender, AsyncCompletedEventArgs e)

{

if (e.Error == null)

{

SetIconStatus(true);

StatusMessage = Resources.SessionInfoSuccess;

UpdateSesionListUi();

UpdateSessionButtons();

}

else

{

SetIconStatus(false);

SessionStatusMessage = Resources.SessionInfoFailed;

DisplayMessageBox(string.Format(Resources.SessionInfoFailed, e.Error));

}

}

private void UpdateSesionListUi()

{

Application.Current.Dispatcher.Invoke(new Action(() =>

{

SessionList.Clear();

foreach (var session in m\_controller.Sessions.OrderByDescending(s => s.Number))

{

SessionList.Add(session);

}

ReportsViewModel.Instance.UpdateReportUIOnStartNewSession();

}));

}