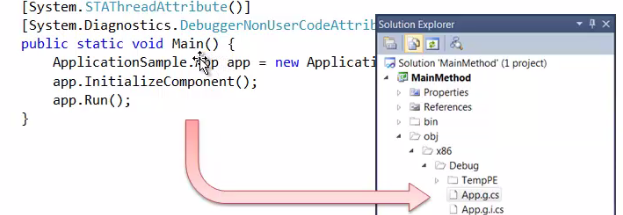
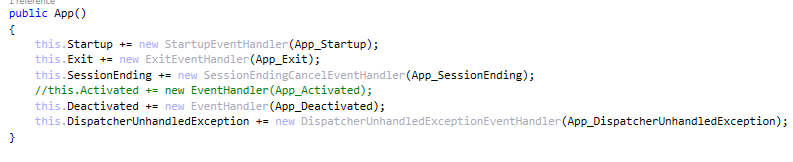
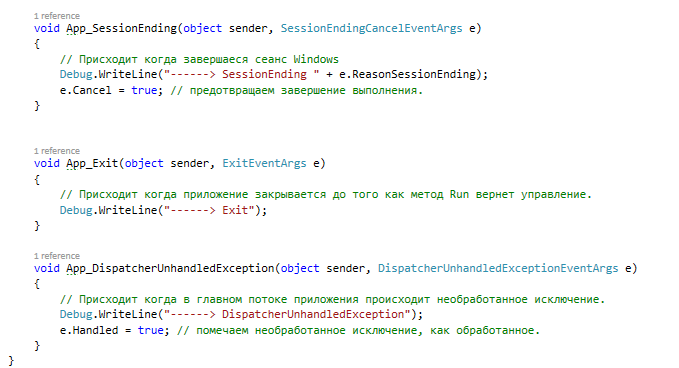
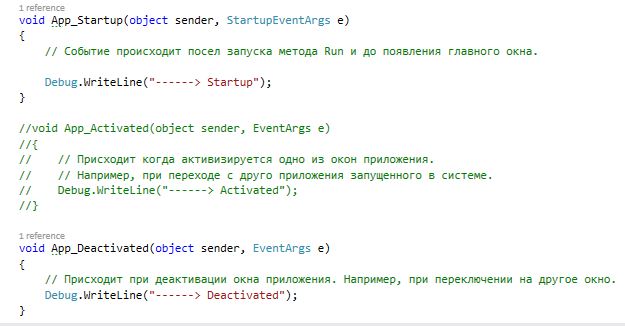
Метод Main()

События Application



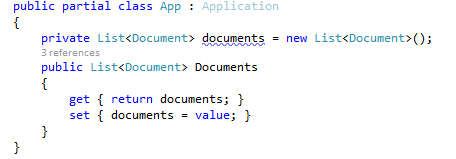


В примере мы добавляем в класс App коллекция в которую добавляем созданные окна

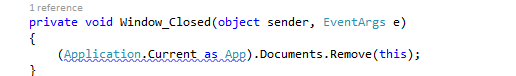
Обновляем их и по закрытию

И удаляем в окна из коллекцию по их закрытию

Создание коллекции (хранение всех дочерних окон)

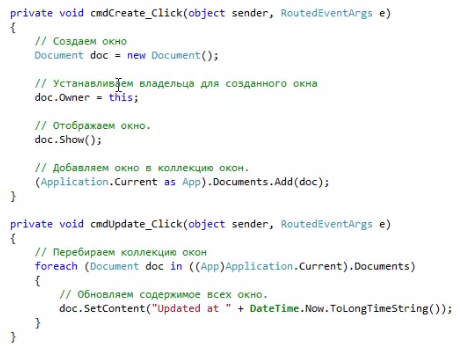


Созданное окно (удаление)

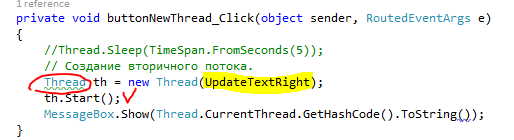


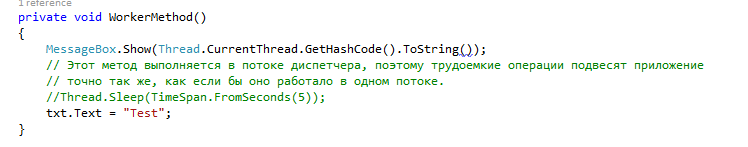
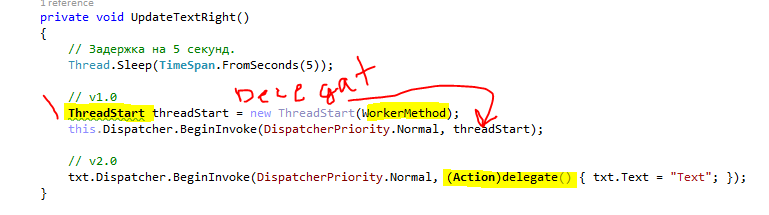
Главное окно (добавление обновление)

В дочернем окне следует указать свойство Owner=this, для того чтобы по закрытию родительского окна дочерние окна так же закрывались

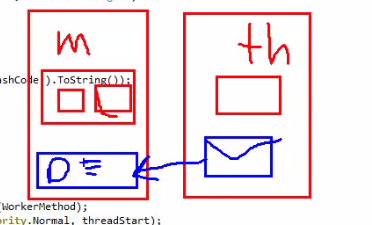


Thread – Потоки





Вторичный поток создает делегат и возвращает его Dispetcher, где по мере выполнения задач (списка который в нем хранится) выполниться и делегат



BackgroundWorker

Для работы необходимо подключить пространство имен



// Метод, который сработает в момент завершения BackgroundWorker

\_worker.RunWorkerCompleted += new RunWorkerCompletedEventHandler(worker\_RunWorkerCompleted);

void worker\_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)

{

this.Title = "Completed";

MessageBox.Show("Completed");

if (e.Cancelled)

this.Title = "Cancelled";

}

// Для отслеживания выполнения хода работ свойство WorkerReportsProgress устанавливаем true

\_worker.WorkerReportsProgress = true;

// Поддержка отмены выполнения фоновой операции с помощью метода CancelAsync()

\_worker.WorkerSupportsCancellation = true;

// Метод, который будет выполнятся в отдельном потоке. Событие DoWork срабатывает при вызове RunWorkerAsync

\_worker.DoWork += new DoWorkEventHandler(worker\_DoWork);

// Данный метод работает в отдельном потоке.

void worker\_DoWork(object sender, DoWorkEventArgs e)

{

for (int i = 0; i <= 100; ++i)

{

// Эмулируем трудоемкую задачу.

Thread.Sleep(50);

// Отмена выполнения фоновой задачи, сработает при вызове CancelAsync

if (\_worker.CancellationPending)

{

e.Cancel = true; // значение нужно установить для того что бы при событии RunWorkerCompleted определить почему оно было вызвано, из-за того что закончилась операция или из-за отмены.

return; // Отмена выполнения фоновой операции.

}

// Отчитываемся о проценте выполнения задачи.

\_worker.ReportProgress(i);

}

}

// Запуск выполнения фоновой операции. Событие DoWork.

// Вторая перегрузка RunWorkerAsync позволяет передать объект событию DoWork для его последующей обработки в потоке.

\_worker.RunWorkerAsync();

Если требуется отменить поток \_worker

// Для работы метода, свойство WorkerSupportsCancellation должно быть равное true.

\_worker.CancelAsync();