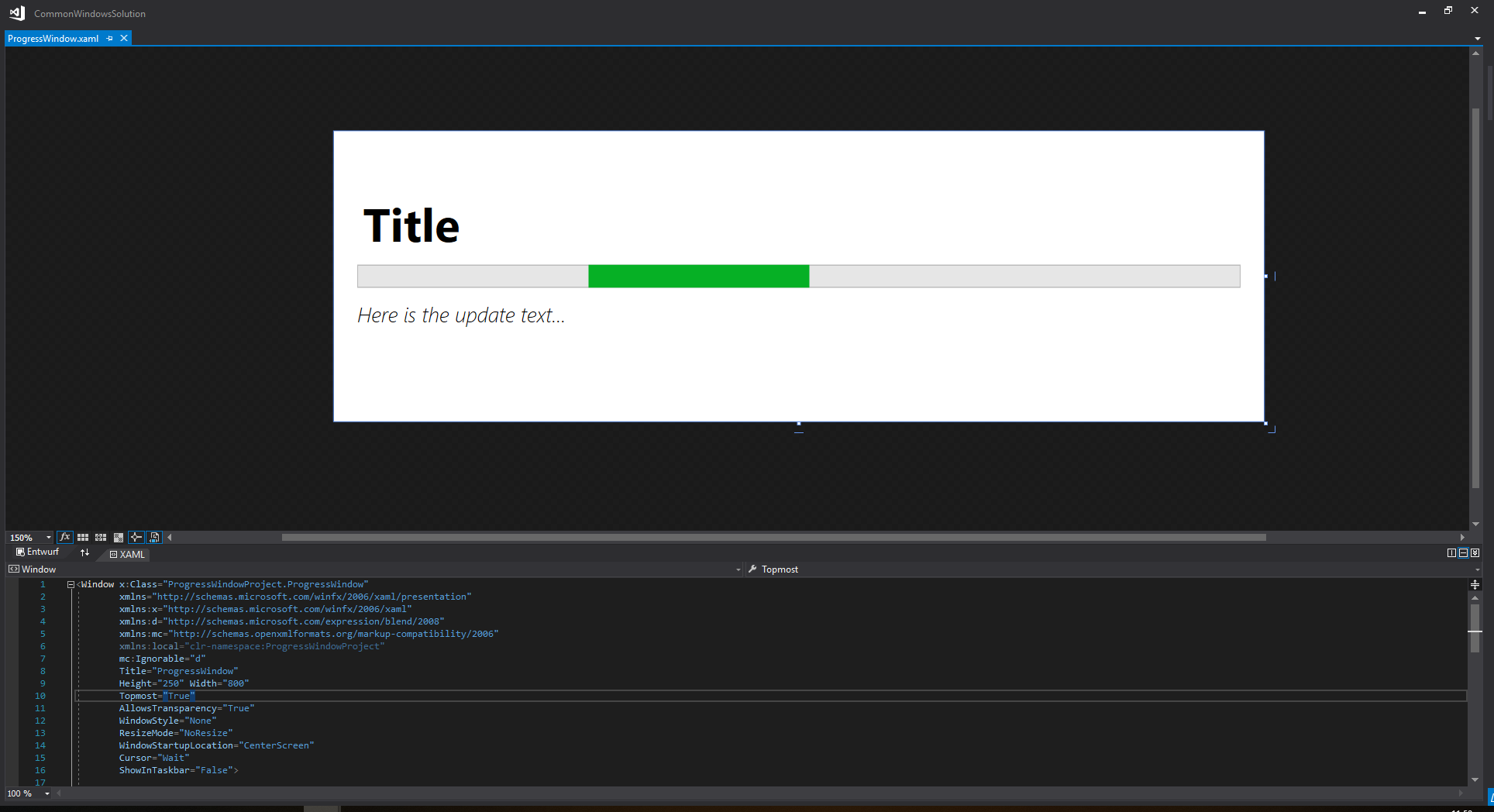
**Progress Window**

**Manual - Lunarix**



**22. Okt. 2018**

Version 1.00

Revision: 1

**Inhaltsverzeichnis**

[1 Projektbeschreibung 3](#_Toc527973198)

[2 Software 4](#_Toc527973199)

[2.1 Technischer Aufbau 4](#_Toc527973200)

[2.2 Einbindung in Projekt 4](#_Toc527973201)

[3 Versionsverlauf 5](#_Toc527973202)

[5 Anhang 6](#_Toc527973203)

[5.1 Glossar 6](#_Toc527973204)

[5.2 Abbildungsverzeichnis 6](#_Toc527973205)

[5.3 Tabellenverzeichnis 6](#_Toc527973206)

[5.4 Formelverzeichnis 6](#_Toc527973207)

# Projektbeschreibung

In heutigen Software-Anwendungen ist man bestrebt, längere Aufgaben im Hintergrund abwickeln zu können. Manchmal kann es aber nicht vermieden werden, dass der User auf den Abschluss einer Aufgabe warten muss.

Das Warten wird aber angenehmer, wenn man über den Fortschritt informiert wird. Zum einen, da man in etwa abschätzen kann, wie lange der Vorgang noch dauern wird und zum anderen, dass man die Gewissheit hat, dass der Rechner am Arbeiten ist.

Um eine unabhängige Anzeige für den Fortschritt in Petto zu haben, ist hier das Progress Window entwickelt worden.

# Software

## Technischer Aufbau

Das Progress Window basiert auf der WPF-Technologie von Microsoft.

Die Schwierigkeit bei der Darstellung einer Fortschrittsanzeige ist, die Anzeige aktuell zu halten (UI-Thread aktualisiert sie periodisch) obwohl nebenbei die eigentliche Aufgabe abgearbeitet werden muss, deren Fortschritt dargestellt werden soll.  
Um diesen Spagat machen zu können, wird auf ein IProgress<ProgressReport>-Objekt zurückgegriffen. Dieses muss sowohl dem ProgressWindow-Objekt als auch dem Task, der die Hintergrundaufgabe ausführt übergeben werden.

Natürlich muss die Darstellung des Progress Window (Methode Show(…) oder ShowDialog(…)) im UI-Thread ausgeführt werden, damit das Fenster aktualisiert wird.  
Dementsprechend muss die eigentliche Aufgabe in einem neuen Task abgearbeitet werden.

## Einbindung in Projekt

Die Einbindung eines UI-Tasks kann so seine Probleme bergen. WPF beschränkt den Zugriff auf UI-Elemente auf einen einzigen Thread. Damit dieser aber arbeiten kann, muss das Projekt richtig eingebunden werden.

### Konsolenapplikation

Bei einer Konsolenapplikation muss der Hauptthread (die Main-Funktion) als statischer Thread gekennzeichnet werden. Dies geschieht durch das Schlüsselwort [STAThread].

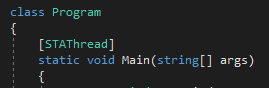


Abbildung 1: Beispiel STAThread.

Nachdem ein ProgressWindow-Objekt und ein IProgress<ProgressReport>-Objekt erstellt ist, wird ein neuer Task mit der abzuarbeitenden Methode gestartet. Dieser erhalt auch das IProgress<ProgressReport>-Objekt als Parameter.

Ist der Task gestartet, kann Show() oder ShowDialog() (intern identisch) auf das ProgressWindow-Objekt aufgerufen werden, damit das Fenster dargestellt wird.  
Da intern auch beim Aufruf der Show()-Methode schlussendlich die ShowDialog()-Methode der Basisklasse Window aufgerufen wird, wartet der UI-Thread an der Aufrufstelle, bis das Fenster geschlossen wird und die Aufgabe somit abgeschlossen oder abgebrochen wurde.

Der Abschluss der Aufgabe kann auf zwei Arten signalisiert werden.

1. Dem IProgress-Objekt wird eine aktuelle Fortschritts-Zahl identisch der totalen Fortschritts-Zahl übergeben.
2. Die Methode Finished() des ProgressWindow wird aufgerufen.

Ein komplettes Beispiel für eine Konsolenapplikation, welche das ProgressWindow einbindet, findet sich im Kapitel 4.1.

### Desktopapplikation

[Beschreiben, sobald gemacht…]

# Versionsverlauf

|  |  |  |
| --- | --- | --- |
| **Version** | **Datum** | **Kommentar** |
| 1.00 | 22.10.2018 | Erstellung des Manuals. |

# Anhang

## Konsolenapplikation – Beispiel

Ein Beispiel für die Implementierung in einer Konsolenanwendung.

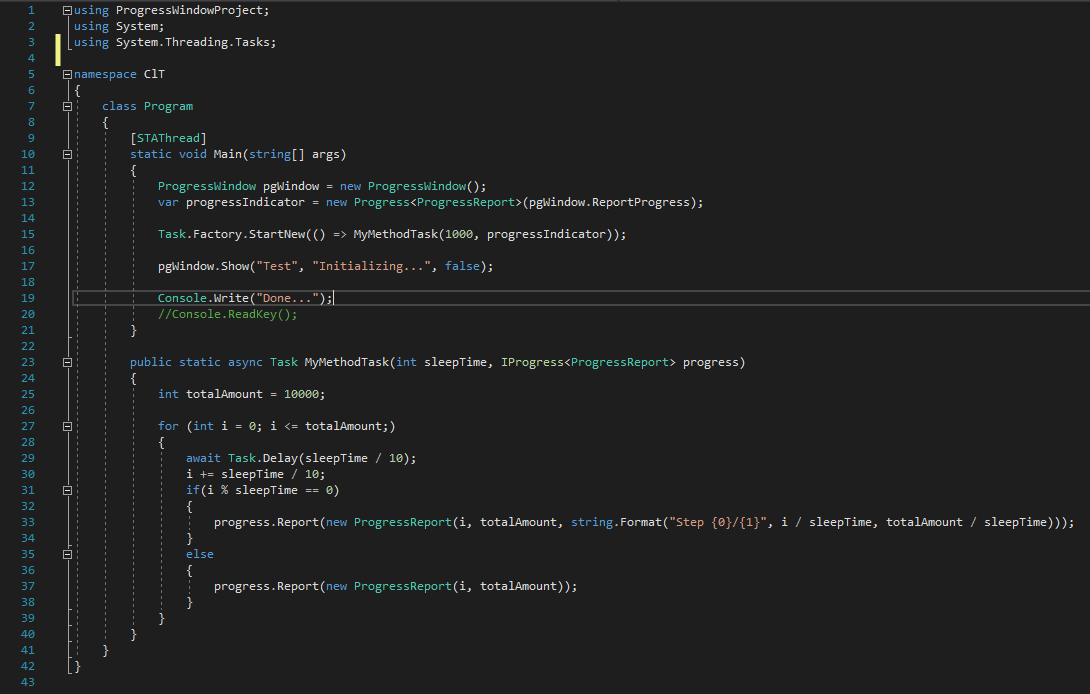


Abbildung 2: Beispiel-Implementation in eine Konsolenanwendung.

## Desktopapplikation – Beispiel

## Abbildungsverzeichnis

[Abbildung 1: Beispiel STAThread. 4](#_Toc527977290)

[Abbildung 2: Beispiel-Implementation in eine Konsolenanwendung. 7](#_Toc527977291)