Catalist – In Quartz.NET via Job.Execute() das UI informieren

Axel Ullrich: 09.02.2016

Hintergrund

Die von Quartz.NET ausgelösten Trigger laufen in eigenen Threads, die nicht ohne weiteres auf den laufenden Thread der Benutzeroberfläche zugreifen können.

Es gibt jedoch die Möglichkeit, Methoden der Benutzeroberfläche mithilfe von Delegates per Invoke() aufzurufen. Allerdings muss dazu die Execute-Methode wissen, welche und wessen Methode sie aufrufen soll.

Der Schlüssel ist die sogenannte *JobDataMap* des *JobDetails* Objekts, dem man beliebig viele Parameter als String-Value Paare übergeben kann. Unter anderem z. B. auch die Instanz eines Formulars, dass eine Methode enthält, die beim Ausführen der *Execute()* Methode aufgerufen werden soll.

Um Abhängigkeiten zwischen der Scheduler Komponente und der Benutzeroberfläche zu vermeiden, findet die Kommunikation nicht über verdrahtete Referenzen sondern über eine Schnittstelle (*IReminderRecipient*) statt. Das ist sinnvoll, damit die Scheduler Komponente keinen direkten Verweis auf die UI Komponente benötigt, um die entsprechende Methode aufzurufen.

Scheduler Komponente

Die Scheduler Komponente ist für das Erstellen des Schedulers und von Jobs und Triggern zuständig. Bei der Job-Erstellung findet die Übergabe des *IReminderRecipient* statt:

Im Scheduler Service Objekt

```
public void AddJob(string description, DateTime startAt, IReminderRecipient caller)
{
  IJob job = new IJob()
    .SetDescription(description)
    .SetStartAt(startAt);

  IJobDetail jobDetail = JobBuilder.Create<IJob>()
    .WithDescription(description)
    .Build();
  jobDetail.JobDataMap["Caller"] = caller
}
```

Im IJob Objekt

```
public void Execute(IJobExecutionContext context)
{
  JobDataMap map = context.JobDetail.JobDataMap;
  IReminderRecipient caller = map["Caller"] as IReminderRecipient;
  if (caller != null)
  {
    caller.HandleReminderFired(context.JobDetail.Description);
  }
}
```

UI Komponente

In der Benutzeroberfläche implementiert das für die Verarbeitung zuständige Formular die *IReminderRecipient.HandleReminderFired()* Methode. Diese Methode ist schließlich dafür verantwortlich, dass die Prozedur aufgerufen wird, die dann die eigentliche „Arbeit“ verrichtet:

```
public void HandleReminderFired(string description)
{
  if (this.InvokeRequired)
  {
    Action action = () => myJob_ReminderFiredEvent(description);
    this.Invoke(action);
    //Oder: this.Invoke((Action)(() => { myJob_ReminderFiredEvent(description); }));
  }
}
```

Die Arbeit findet hier statt:

```
void myJob_ReminderFiredEvent(string description)
{
  string msg = string.Format("Erinnerung: {0}!", description);
  MessageBox.Show(msg, "Alaaaaarm", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```