Task Parallel Library (TPL)
Klasse Task – Teil 3

Ingo Köster

Diplom Informatiker (FH)

Fortsetzungen von Tasks

- Tasks können durch andere Tasks fortgesetzt werden
- Wird eine Aufgabe in zwei oder mehr Teile zerlegt, ist es ggf. notwendig nach Beendigung einer Aufgabe eine oder mehrere andere Aufgaben zu starten
- Es ist nicht notwendig auf die Beendigung eines Task zu warten, ein weiterer Task kann als Fortsetzung definiert werden
- Die Fortsetzung startet, sobald der der ursprüngliche Task beendet wurde

Fortsetzung - ContinueWith

```
Console.WriteLine("Starte einen Task...");

Task taskMitFortsetzung = Task.Factory.StartNew(() =>
Thread.Sleep(3000));

Console.WriteLine("Task {0} wurde gestartet...",
taskMitFortsetzung.Id);

taskMitFortsetzung.ContinueWith(vorherigerTask =>
  Console.WriteLine("Die Fortsetzung von Task {0}",
vorherigerTask.Id));
```

Fortsetzung - `ContinueWith`

- Ausgabe des Beispiels:
  - `Starte einen Task...`
  - `Task 1 wurde gestartet...`
  - `Die Fortsetzung von Task 1`

- Der Fortsetzung wird der vorherige Task als Parameter übergeben

- Es können mehrere `ContinueWith` Anweisungen an einen Task angehängt werden

- Auch eine Fortsetzung kann wiederum fortgesetzt werden

ContinueWith Überladungen

› `ContinueWith` ist überladen

› Mit `TaskContinuationOptions` kann angegeben werden unter welchen Umständen eine Fortsetzung stattfinden soll
  › Z.B. `NotOnCanceled, NotOnFaulted, OnlyOnCanceled, OnlyOnFaulted, etc.`

```
Task<int> einTask = Task.Run(() => { … });
einTask.ContinueWith((vorheriger) => { … },
 TaskContinuationOptions.OnlyOnRanToCompletion);
```

Kind-Tasks

- In einem Task können Kind-Tasks angelegt werden
- Der Eltern-Task endet erst, wenn alle Kind-Tasks beendet wurden

```
Task elternTask = Task.Factory.StartNew(() =>
{
  new Task(() => { … },
 TaskCreationOptions.AttachedToParent).Start();
  new Task(() => { … },
 TaskCreationOptions.AttachedToParent).Start();
});
```

- Mittels der Klasse `TaskFactory` können Tasks mit gleicher Konfiguration erstellt werden

Kind-Tasks

Kategorie	Getrennter untergeordneter Task	Mit `TaskCreationOptions.AttachedToParent`
Eltern-Task wartet auf das Ende der Kind-Tasks	Nein	Ja
Eltern-Task gibt von Kind-Tasks ausgelöste Ausnahmen weiter	Nein	Ja
Der Status (Property `Status`) des Eltern Task hängt vom Status der Kind-Tasks ab	Nein	Ja