bib International College

Threads

Teil 5

Dateioperationen mit FileStream

- **Synchrone Leseoperation**:

public override int Read(byte[] *array*, int *offset*, int *count*)

- **Aysnchrone Leseoperation**:

public override IAsyncResult BeginRead(
byte[] *array*,
int *offset*,
int *numBytes*,
AsyncCallback *userCallback*,
object *stateObject*)

public override int EndRead(IAsyncResult *asyncResult*)

Dateioperationen mit FileStream

- **Synchrone Schreiboperation**:

public override void Write(byte[] *array*, int *offset*, int *count*)

- **Aysnchrone Schreiboperation**:

public override IAsyncResult BeginWrite(
byte[] *array*,
int *offset*,
int *numBytes*,
AsyncCallback *userCallback*,
object *stateObject*)

public override void EndWrite(IAsyncResult *asyncResult*)

Callback-Methoden

bib International College

- Asynchrone Methodenaufrufe in .NET direkt mit Delegates implementieren → Callback-Mechanismus
- wenn ein Delegate erzeugt wird, generiert der Compiler drei Methoden**: Invoke**, **BeginInvoke** und **EndInvoke**.
- **Invoke** führt synchron aus
- **BeginInvoke** und **EndInvoke** zur asynchronen Methoden-Ausführung
- jedem Aufruf von BeginInvoke muss ein EndInvoke folgen
- BeginInvoke übernimmt dieselben Argumente, wie der Delegate und zusätzlich noch zwei weitere Parameter - einen Delegate vom Typ ASyncCallback, und das eigene Delegate
- AsyncCallback ist eine Delegate für eine Methode, die void liefert und ein einziges Argument hat, nämlich ein Objekt des Typs IAsyncResult.

BeginInvoke

bib International College

Ein Aufruf der Methode BeginInvoke auf der Referenz eines Delegaten erzeugt einen Hintergrundthread, in welchem die vom Delegaten beschriebene Methode ausgeführt wird. Die Parameter für diese Methode werden BeginInvoke übergeben. Der aufrufende Thread arbeitet weiter, anstatt auf die Beendigung der aufgerufenen Methode zu warten.

```
public delegate void MyDelegate();
MyDelegate del = new MyDelegate(obj.DoSomething);
del.BeginInvoke(…);
```

Member von System.ComponentModel.ISynchronizeInvoke:

```
public IAsyncResult BeginInvoke( [Parameterliste,]
                                 System.Delegate method,
                                 Object args)
```

Beendigung des Hintergrundthreads

I.d.R. will der aufrufende Code über die Beendigung der asynchronen Methodenausführung in Kenntnis gesetzt werden, damit er die Rückgabewerte verarbeiten kann.

Die asynchron aufgerufene Methode muss den Aufrufer darüber informieren, also muss ihr die Adresse der Rückrufmethode im Aufrufer bekannt sein.

Dazu wird dem Aufruf von BeginInvoke zusätzlich zu der Parameterliste noch das erforderliche Delegate auf die Rückrufmethode übergeben.

Definition des Delegates AsyncCallback:

public delegate void AsyncCallback (IAsyncResult ar)

Die Rückrufmethode muss den Rückgabetyp void aufweisen und einen Parameter von Typ IAsyncResult definieren.

Asynchroner Methodenaufruf ohne Rückgabeparameter

```
class Demo{
        public void DoSomething() {
            for (int i = 0; i <= 30; i++) {
                Console.Write("X");
                Thread.Sleep(10);
            }
        }
}
```

```
class Program {
    public delegate void MyDelegate();
    private static MyDelegate del;
    static void Main(string[] args) {
        Demo d = new Demo();
        del = new MyDelegate(d.DoSomething);
      AsyncCallback cb = new AsyncCallback(RückrufMethode);
      del.BeginInvoke(cb, null);
      for (int i = 0; i <= 100; i++) {
          Console.Write("."); Thread.Sleep(10);
      }
      Console.ReadLine();  Console.WriteLine("Ende");
   }

  public static void RückrufMethode(IAsyncResult ar) {
      Console.Write("fertig");
  }
}
```

Das Objekt vom Typ IAsyncResult entspricht dem Rückgabewert von BeginInvoke

Asynchroner Methodenaufruf mit Rückgabeparameter

```
class Demo{
        public string DoSomething() {
            for (int i = 0; i <= 30; i++) {
                Console.Write("X");
                Thread.Sleep(10);
            }
            return "fertig";
        }
}
```

EndInvoke

Die asynchrone Methode liefert einen Rückgabewert.

Wird aus dem Hintergrundthread die Rückrufmethode des Initiators der asynchronen Operation aufgerufen, muss mittels der Methode EndInvoke des Delegates der Rückgabewert explizit abgerufen werden.

public Object EndInvoke ([Parameterliste,]
IAsyncResult *result*)

Achtung: Die Parameterliste ist nicht identisch zu der Parameterliste von BeginInvoke. Sie darf nur die Referenzparameter der asynchronen Methode enthalten.

```
class Program {
    public delegate String MyDelegate();
    private static MyDelegate del;
    static void Main(string[] args) {
        Demo d = new Demo();
        del = new MyDelegate(d.DoSomething);
      AsyncCallback cb = new AsyncCallback(RückrufMethode);
      del.BeginInvoke(cb, null);
      for (int i = 0; i <= 100; i++) {
          Console.Write("."); Thread.Sleep(10);
      }
      Console.ReadLine();  Console.WriteLine("Ende");
   }

  public static void RückrufMethode(IAsyncResult ar) {
      String s = del.EndInvoke(ar);
      Console.Write(s);
  }
}
```

Das Objekt vom Typ IAsyncResult entspricht dem Rückgabewert von BeginInvoke

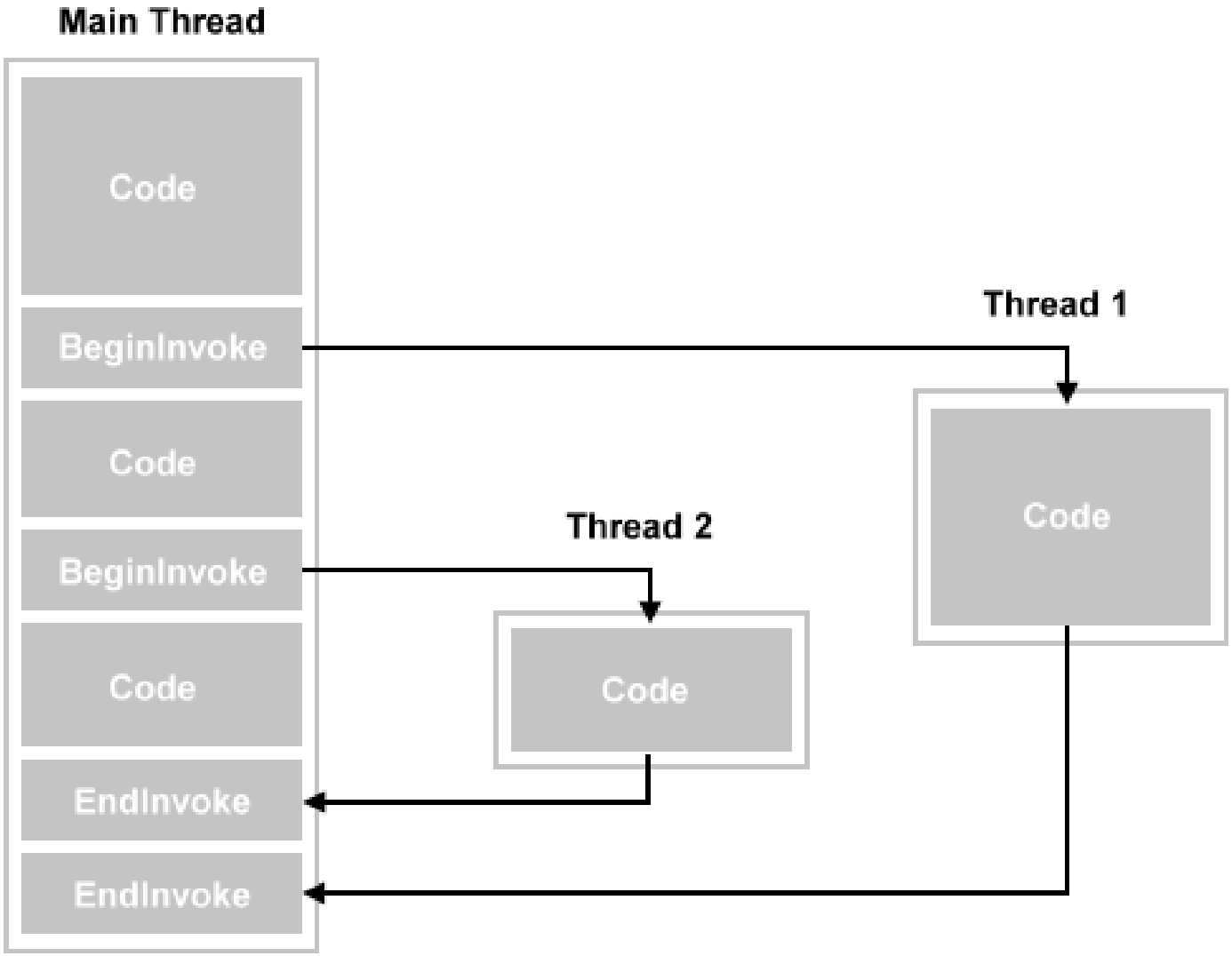
weiteres Beispiel

```
class CallbackTest {
    public static void Main(string[] args) {
        ClassWithDelegate cwd = new ClassWithDelegate();
        Teilnehmer1 t1 = new Teilnehmer1();
        t1.Starte(cwd);
        Teilnehmer2 t2 = new Teilnehmer2();
        t2.Starte(cwd);
        cwd.Run();
    }
}
```

```
public class ClassWithDelegate  {
    // Ein Multicast-Delegate (= Delegate mit mehr als einer Methode in der Aufrufliste) ,
    // dessen gekapselte Methode ein int liefert
    public delegate int DelegateLiefertZahl();
    public event DelegateLiefertZahl delegateZahl;
    public void Run() {
        while(true) {
            Thread.Sleep(500);   // Eine halbe Sekunde schlafen
            if (delegateZahl != null) {
                // Alle delegierten Methoden explizit aufrufen
                 foreach (DelegateLiefertZahl del in delegateZahl.GetInvocationList()) {
                     // Asynchron aufrufen, das Delegate als Status-Objekt übergeben
                     del.BeginInvoke(new AsyncCallback(ErgebnisZurückgeben), del);
     } } } }
    // Callback-Methode zum Abfangen der Ergebnisse
    public void ErgebnisZurückgeben(IAsyncResult iar) {
        // Das Status-Objekt zurück in den Delegate-Typ wandeln
        DelegateLiefertZahl del = (DelegateLiefertZahl)iar.AsyncState;
        // Beim Delegate EndInvoke aufrufen, um das Ergebnis zu bekommen
        int result = del.EndInvoke(iar);
        Console.WriteLine("Delegate liefert Ergebnis: {0}", result);    // Ergebnis
    }                                                                    // anzeigen
}
```

```
public class Teilnehmer1 {
        private int zähler = 0;
        public void Starte(ClassWithDelegate o)  {
              o.delegateZahl += new ClassWithDelegate.DelegateLiefertZahl(BerechneZahl);
        }
       public int BerechneZahl() {
              Console.WriteLine("Teilnehmer1 beschäftigt...");
              Thread.Sleep(10000);
              Console.WriteLine("Teilnehmer1 fertig...");
             return   ++zähler;
       }
 }

 public class Teilnehmer2 {
       private int zähler = 0;
       public void Starte(ClassWithDelegate o){
            o.delegateZahl += new ClassWithDelegate.DelegateLiefertZahl(Verdopple);
        }
        public int Verdopple() {
           return zähler += 2;
        }
   }
}
```


Main Thread
Code
BeginInvoke
Code
BeginInvoke
Code
EndInvoke
EndInvoke
Thread 1
Code
Thread 2
Code