.net column

COM+ in .NET applicatie architectuur

Met de introductie van Windows 2000 maakte ook de nieuwe applicatie server COM+ zijn intrede. Het doel van COM+ was om enerzijds een schaalbaar multi-threaded proces te bieden waar COM dll's in gehost kunnen worden en anderzijds om via declaratieve attributen gebruik te maken van complexe systeem services. COM+ ondersteunt de volgende services: Automatische transacties, synchronisatie, just-in-time activation, object pooling, queued components, compensation resource manager en loosely coupled events.

Om vanuit .NET code gebruik te kunnen maken van deze services dient men een zogenaamd *Serviced Component* te maken. Een serviced component is feitelijk een gewone .NET klasse die afleidt van de *ServicedComponent* basis-klasse, waarin de mapping tussen managed- en unmanaged code wordt geregeld. Daarnaast zijn er vele .NET attributen beschikbaar die gebruikt worden om de gewenste COM+ services op een juiste wijze te configureren. Technisch is het dus mogelijk om vrijwel alle COM+ services vanuit een .NET assembly te gebruiken. Ondanks dat er geen grote technische beperkingen zijn moet men erg kritisch zijn over het gebruik van COM+ vanuit een .NET architectuur.

Er zijn diverse redenen aan te geven om terughoudend te zijn in het gebruik van COM+ vanuit een .NET architectuur:

In de tijd dat de Microsoft Transaction Server (MTS = voorganger van COM+) en COM+ ontwikkeld zijn was de COM technologie erg belangrijk voor Microsoft en bestond .NET nog niet. Men wilde het bouwen van COM-componenten eenvoudig maken door aan diverse programmeertalen krachtige COM ondersteuning toe te voegen. Zo werd Visual Basic de belangrijkste taal voor het ontwikkelen van COM-componenten. Echter, door het ontbreken van multi-threading in (klassiek) Visual Basic, was het niet echt goed mogelijk om een schaalbare COM-server te bouwen. Om dit probleem op te lossen was een applicatieserver benodigd, die via een pool van (STA) threads, voldoende schaalbaar was om vele verzoeken gelijktijdig af te kunnen handelen.

In .NET is multi-threading en synchronisatie geen issue meer. Het creëren van nieuwe threads is kinderspel geworden en voor de benodigde synchronisatie van code is er rechtstreekse ondersteuning vanuit de .NET programmeertalen (C#: lock, VB.NET: *SyncLock* enz.)

Verder moet men zich goed realiseren dat voor de remoting van COM+ serviced components (.NET assembly's) er 'gewoon' gebruik wordt gemaakt van het DCOM protocol. In een moderne .NET architectuur, veelal gebaseerd op internet technologie (IIS + ASP.NET, Firewalls, DMZ, enz.) levert DCOM vaak veel problemen op.

Deze problemen hebben te maken met de keuze voor een disconnected of asynchrone manier van werken, waar DCOM niet geschikt voor is of men loopt tegen firewall problemen aan, die veroorzaakt worden door de grote reeks aan benodigde udp-poorten voor de onderliggende RPC verbinding.

De enige service van COM+, waarvoor er in .NET geen alternatief is en die nuttig is om in een serieus project in te zetten is de ondersteuning van gedistribueerde transacties. Een gedistribueerde transactie is een transactie waaraan meerdere resource managers (bijv. Database servers) deelnemen. Voor lokale transacties (een resource manager) moet zeker geen COM+ worden ingezet, omdat COM+ alle transacties via de DTC laat verlopen, wat behoorlijk wat overhead introduceert.

Via de *System.EnterpriseServices* namespace wordt het zeer eenvoudig om vanuit een .NET assembly transparant gebruikt te maken van de COM+ services. Toch past COM+ vaak niet in een moderne .NET architectuur en is de noodzaak ervan grotendeels overbodig geworden. Een uitzondering hierop is de ondersteuning van gedistribueerde transacties, waarvoor er in .NET geen goed alternatief is.

Ing. Xander Buffart is werkzaam als IT-architect bij Info Support te Veenendaal (e-mail: xanderb@infosupport.com)