.net column

Code achter code-behind pagina

Een Button op een Webform wordt door Visual Studio .NET standaard als protected gedeclareerd, terwijl een Button op een Winform standaard private is. Waarom is dat verschil er eigenlijk?

Voordat we die vraag goed kunnen beantwoorden moeten we eerst weten hoe ASP.NET omgaat met z'n web pagina's. Twee verschillen met 'traditionele' ASP pagina's springen daarbij in het oog. Belangrijk is dat de ASP.NET pagina's worden gecompileerd en dus niet langer worden geïnterpreteerd, zoals dat bij 'traditionele' ASP pagina's wel het geval was. Dit verhoogt de efficiëntie enorm. Verder is er een meer strikte scheiding aangebracht tussen code en presentatie. Niet langer staan script en HTML zomaar door elkaar; de presentatie van de pagina wordt geregeld in de (pseudo-)HTML op de aspx-pagina zelf, terwijl de code in een aparte code-behind pagina wordt geplaatst. Dit heeft naast overzichtelijkheid ook het grote voordeel dat webdesigner en .NET-onwikkelaar hun taken relatief onafhankelijk van elkaar kunnen uitvoeren. Een webdesigner kan de opmaak van een pagina wijzigen zonder zelf opnieuw iets te hoeven compileren.

Om af te zijn van interpretatie maar tegelijkertijd webdesigners niet lastig te hoeven vallen met compilers, is er in ASP.NET voor gekozen om aan elke aspx-pagina een assembly te koppelen. Deze assembly wordt door de asp.net-engine automatisch gegenereerd als een veranderde aspx-pagina op de webserver voor het eerst wordt opgevraagd. In die assembly zit een klasse die op verzoek van een client (browser) precies die HTML produceert die nodig is om de pagina in de browser correct weer te geven.

Dat is ingewikkelder dan het lijkt, want om de correcte html-pagina over de lijn te kunnen sturen, moet natuurlijk ook alle code van de code-behind pagina uitgevoerd zijn. Om al die zogenaamde 'server-side events' op een eenvoudige manier beschikbaar te maken, wordt de klasse die verantwoordelijk is voor het produceren van HTML afgeleid van de klasse in de code-behind pagina.

En dat is precies de crux: webcontrols worden geïnstantieerd in de (afgeleide) klasse die door de ASP.NET engine wordt gegenereerd, maar hun functionaliteit moet ook aangesproken kunnen worden in basis klasse op de code-behind page. Daarom worden de webcontrols gedeclareerd in klasse op de code-behind page, met protected access.

De koppeling tussen de variable op de code-behind pagina en die in de gegenereerde klasse wordt gemaakt op basis van naam. De variable op de code-behind pagina moet exact dezelfde naam hebben als het ID van de bijbehorende tag op de aspx-pagina. Komen deze namen niet overeen of heeft de variabele op de code-behind pagina geen protected maar private access gekregen, dan zal de ASP.NET-engine geen verband leggen. Er zal dan in de gegenereerde klasse een extra variable worden gedecrareerd, met als naam de ID van de tag. Deze extra variable zal een referentie zijn naar de echte instantie van het webcontrol. De variable op de code-behind pagina zal ongeïnitialiseerd blijven en eventuele events de daaraan gekoppeld worden, zullen nooit afgaan.

Deze situatie zal overigens met de komst van *Whidbey* enigszins veranderen. Vanaf Whidbey is het in .NET mogelijk om partiële klassen te definiëren: klasses waarvan de definitie over meer files verspreid is. Deze techniek zal ook in ASP.NET gebruikt gaan worden. De gegenereerde klasse en de code-behind pagina zullen allebei delen van dezelfde klasse vormen. De koppeling op basis van naam zal wel blijven bestaan.

Marco Pil is trainer / consultant bij Info Support