De daily build

Een daily buildproces maakt jouw ontwikkelproces betrouwbaarder. Door minstens elke dag/nacht een geautomatiseerde build met integratietests uit te voeren, kunnen bugs en integratieproblemen sneller worden opgemerkt. Bovendien geeft een daily buildproces inzicht in de voortgang van een project en heb je altijd een versie beschikbaar die klaar is om te testen of potentieel op te leveren. Op deze manier voorkom je allerlei onaangename en vertragende verrassingen in de integratiefase, zoals niet-compilerende code en niet-werkende setups.

Microsoft Team Foundation Server bevat een Build Engine die gebaseerd is op MSBuild. Met behulp van een wizard kan eenvoudig een MSBuild-script gemaakt worden. Er is echter veel meer mogelijk met de Team Foundation Build Server dan wat de BuildType-wizard doet vermoeden. Als je een build maakt, kun je in de wizard selecteren welke solutions je wilt builden en welke build-configuraties daarbij gebouwd moeten worden. In Visual Studio worden bij een nieuwe solution standaard twee buildconfiguraties aangemaakt, namelijk de debug- en release-configuratie. Een debug-build wordt vaak door de ontwikkelaars gebruikt, terwijl de release-build (zonder debugging-files en met compileroptimalisaties) gebruikt wordt door de testers en op de productieomgeving.

Het is aan te raden om zowel een debug- als een release-build te maken. Op die manier kun je als ontwikkelaar, als er op de testomgeving of in productie problemen zijn, altijd op dezelfde code debuggen. Bovendien is het aan te raden om voor een release-build ook de PDB-bestanden te genereren. Deze mogen uiteraard niet meegeleverd worden, maar op deze manier heb je in ieder geval de mogelijkheid om de versie die op productie staat te debuggen.

De standaard build haalt de source-code op uit source-control, compileert deze, plaatst een label op de source-code en kopieert de resulterende binaries naar de outputlocatie. MSBuild biedt de mogelijkheid voor en na elke stap custom-actions uit te voeren. Zo kun je bijvoorbeeld vlak voor het compileren ingrijpen om er voor te zorgen dat elke build een uniek versienummer krijgt. Hiervoor heeft het MSBuild-team de AssemblyInfoTask ontwikkeld. Deze custom buildtask maakt het mogelijk om vrijwel alle instellingen in de AssemblyInfo-files aan te passen. Deze blogpost beschrijft waar je de assemblyinfotask kunt downloaden en hoe je deze kunt toepassen: http://blogs.msdn.com/msbuild/archive/2005/11/11/491947.aspx

Het is vrij eenvoudig om er voor te zorgen dat assemblies tijdens het compileren direct worden gesigned. Signing is een manier om te bepalen of een assembly (applicatie en DLL) gemaakt is door een bepaalde uitgever en dat deze ongewijzigd is. Hiermee kun je bijvoorbeeld afdwingen dat alleen code die gesigned is met key X uitgevoerd kan worden op de computers van eindgebruikers. Vooral voor applicaties die via intra/internet worden verspreid, kan dit handig zijn. Normaal worden deze uitgevoerd met beperkte rechten (in de intra/internetzone). Met behulp van policies is het mogelijk om aan assemblies van een vertrouwde uitgever extra rechten toe te kennen. Aangezien je niet wilt dat iedereen zomaar code kan signen met de key van jouw bedrijf, wordt het op een veilige manier omgaan met de keys of certificaten een stuk belangrijker. Een daily build kan je hiermee helpen. De buildserver kan namelijk de signing uitvoeren.

Sla de key op een veilige plaats op de buildserver op, zodat alleen het TFS Serverproces erbij kan. Zorg uiteraard wel voor adequate beveiliging van de buildserver. In de Response-file kun je aangeven dat de resultaten van de build gesigned moeten worden met de key van jouw bedrijf. Deze blogpost beschrijft hoe dit werkt: http://blogs.msdn.com/msbuild/archive/2005/09/26/474079.aspx Ontwikkelaars zouden dus geen toegang mogen hebben tot de private keys. Mocht je op de ontwikkelomgeving toch willen testen met gesignde assemblies, dan kun je gebruikmaken van delayed signing. Hierbij sign je de assemblies met de public key, in plaats van met de private key. Dit is uiteraard geen volwaardige signing, maar je kunt jouw ontwikkelomgeving foppen door aan te geven dat delayed signed assemblies ook worden aangezien als gesignde assemblies.

Na het compileren wil je controleren of het resultaat van de build voldoet aan de gestelde kwaliteitseisen. Bij het maken van een nieuwe BuildType zal de wizard vragen welke unittests de build moet uitvoeren. Hiermee kunnen de BVT's (Build Verification Tests) prima uitgevoerd worden. De buildserver is echter geen representatieve testomgeving en het installatieproces wordt bijvoorbeeld niet getest.

Het is mogelijk om de buildresultaten automatisch te laten installeren op een testomgeving. De onderstaande blog beschrijft vrij duidelijk hoe dit werkt: http://lvildosola.blogspot.com/2007/05/automatically-launch-silent-remote-msi.html

Als je de deployment naar de testserver automatisch laat uitvoeren, heeft het testteam altijd de beschikking over de laatste versie van het systeem. Dit kan erg handig zijn als de testers ook meedraaien in het projectteam. Dit neemt echter niet weg dat er ook een stabiele testomgeving moet zijn, waar één bepaalde release geïnstalleerd staat. Soms zal het mogelijk zijn om deze parallel aan de daily build te installeren op dezelfde testomgeving, maar het is netter om hiervoor een aparte testomgeving voor te realiseren.

Het virtualiseren van de testomgeving maakt ook nog een aantal interessante scenario's mogelijk. Elke build kan bijvoorbeeld automatisch gedeployed en getest worden op een nieuwe, schone testomgeving. Als er behoefte is aan een stabiele omgeving, kan een kloon van de testomgeving van een bepaalde versie gemaakt worden. Ook kan een testomgeving bewaard worden op het moment dat een systeem in productie gaat, zodat eventuele problemen in productie na te spelen zijn op de testomgeving.

De installatieprogramma's die je met Visual Studio kunt maken, worden helaas niet standaard ondersteund in de Team Foundation Build. Het is mogelijk om Visual Studio ook op de buildserver te installeren en via de commandline Visual Studio de setup te laten generen. Maar omdat je waarschijnlijk geen Visual Studio op de buildserver wilt hebben, is het maken van een installatieprogramma met WIX (Windows Installeert Xml) een mooiere oplossing. Het onderstaande MSDN-artikel beschrijft hoe dit werkt: http://msdn.microsoft.com/msdnmag/issues/07/03/WixTricks/

'Uit de praktijk' is een column van Clemens Schotte en Erwin van der Valk. Beiden zijn werkzaam als Development Consultant bij Microsoft Services.