Versiebeheer met Team Foundation Server 2010

Marcel de Vries

Veel ontwikkelaars weten het al, maar werken zonder een versie beheer pakket in teamverband betekent chaos. Daarom is het een van de tools waar een ontwikkelaar eigenlijk niet zonder kan. Al sinds jaar en dag levert Microsoft versie beheer in de vorm van Visual Source Safe. Sinds 2005 is versie beheer ook mogelijk met het product Team foundation Server (TFS).

Beide versie beheer pakketten zijn in de basis erg simpel te gebruiken. Wel is er een belangrijk aantal verschillen tussen beide pakketen. Nu dat met de release van Visual studio 2010, het product visual Source Safe officieel aan het einde van zijn life cycle is gekomen en vanaf nu ook niet meer te verkrijgen is als product, is dit het moment om kennis te nemen van deze verschillen en te zien hoe je TFS kunt inzetten. In dit artikel wil ik een ieder die nog niet zo bekend is met TFS, uitleggen wat de terminologie is die wordt gebruikt in TFS en een aantal basis concepten uit de doeken doen. Daarnaast zal ik op een aantal punten wijzen waarin TFS duidelijk een andere benadering heeft voor versie beheer ten opzichte van Source Safe.

Server product versus een netwerk share

Zoals de naam van het product al doet vermoeden is TFS ook daadwerkelijk een server product die als aparte server in je netwerk beschikbaar moet zijn. Bij Visual Source Safe was er sprake van een gemeenschappelijke netwerkshare waar iedere Source Safe client de bronbestanden zelf op de juiste manier neerzette. Met TFS wordt er door middel van een Client applicatie gecommuniceerd met een server, die op zijn beurt verantwoordelijk is voor het op de juiste manier opslaan van de sources. TFS maakt hiervoor gebruik van SQL server om de sources op te slaan.
De communicatie tussen de client en de server vindt plaat op basis van het webservices protocol (SOAP). Bij het benaderen van de server wordt gebruik gemaakt van integrated security, dat betekend dat als de TFS server in het zelfde domein staat, je automatisch verbinding kunt krijgen met de server op basis van je domein gebruikersaccount. Dit betekent ook dat je dus niet meer door een andere gebruikersnaam op te geven zomaar als een andere gebruiker de TFS server kunt benaderen.
Er wordt gebruik gemaakt van active directory gebruikersaccount welke in TFS bepaalde rechten geeft. Op de server wordt beveiliging afgedwongen, dus je hebt alleen toegang tot delen van de sources waar de TFS beheerder rechten voor heeft uitgedeeld.

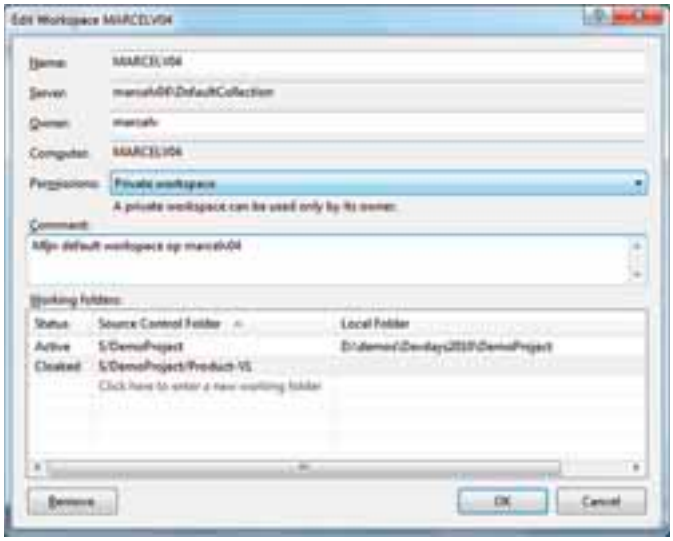
Schaalbaarheid

Belangrijk aan de nieuwe 2010 release van Team Foundation Server zijn onder meer de nieuwe opties als het gaat om de installatie van het product. Microsoft heeft erg veel energie gestopt om de Team Foundation Server dusdanig schaalbaar te maken dat je deze prima kunt gebruiken voor versie beheer op je eigen persoonlijke laptop maar ook voor de enterprise met honderden gebruikers. Daarom is de installatie ook sterk vereenvoudigd en is de server nu in te richten opeen aantal verschillende manieren. Voor individueel gebruik is een basic installatie voldoende, waarbij naast versie beheer ook de build en work item tracking omgeving beschikbaar is. Voor kleine organisaties is een standaard installatie veelaal voldoende waarbij één server wordt ingericht als de centrale team foundation server, waarbij version control, build, work item tracking, reporting en het portaal dan vanaf de centrale TFS beschikbaar zijn. In een enterprise scenario is het mogelijk de TFS server in te richten en hierbij meerdere zogenaamde applicatie tiers en data tiers in te richten om op die manier maximale schaalbaarheid maar ook fail over te realiseren.

Workspaces

Om de sources lokaal te kunnen bewerken, moet er een zogenaamde Workspace worden aangemaakt. Dit zal Visual Studio normaal gesproken automatisch voor je doen als je de eerste keer verbinding maakt met de server, maar het is wel belangrijk om het concept van een workspace toch nader te belichten. Een workspace wordt geadministreerd op de server en bevat alle informatie over welke bronbestanden je als ontwikkelaar lokaal op je machine hebt staan, welke status deze bestanden hebben en welke versie van de bestanden lokaal bij je op de machine staan. Verder bevat een workspace de mapping tussen bestanden in de version control server en je lokale bestands systeem. Een mapping geeft aan welke locatie in version control op welke locatie op je lokale bestandssysteem moet worden geplaatst. In een workspace kun je meerdere van dit soort mappings opnemen, maar het is aan te bevelen er maar een te maken op een zo hoog mogelijk niveau in de versiebeheer boomstructuur op de server. Zodra je in de version control explorer een commando geeft om de laatste sources op te halen, dan zal op de server worden gekeken welke bestanden je al eerder hebt opgehaald en welke bestanden in de tussentijd zijn veranderd. Alleen bestanden die zijn veranderd zullen vervolgens worden opgehaald, zodat de bandbreedte zo optimaal mogelijk wordt gebruikt.
In Figuur 1 is de dialoog te zien voor het definiëren van een workspace. Hierin is te zien welke mapping is aangegeven voor de

workspace marcelv04. Zoals je kunt zien in de dialoog, is de workspace gebonden aan een gebruiker en een machine. Voor iedere ontwikkelmachine waar je op werkt dien je een workspace aan te maken. Verder zie je dat er zogenaamde permissies kunnen worden gezet op een workspace. Daarbij kent men drie varianten te weten: private, beperkte public de volledige public workspace. Private betekend dat de workspace voor deze ontwikkelaars is op deze machine en niet kan worden gebruikt door een andere ontwikkelaar op deze machine. Beperkt public betekend dat één ontwikkelaar de workspace aanmaakt en een andere ontwikkelaar wel de workspace mag gebruiken maar niet zelf mag aanpassen. Bij een volledig public workspace, mag een ieder op de machine ook de workspace aanpassen. Verder is het mogelijk dat je als ontwikkelaar meerdere workspaces gebruikt. Het werken met meerdere workspaces is in de meeste gevallen niet nodig. Meerdere workspaces zijn met name handig als je ook met meerdere team projecten tegelijk te maken hebt. Meestal werk je aan een project tegelijk en wil je ook alleen wijzigingen zien die van toepassing zijn op dat specifieke project. Bij het inchecken van de sources, zie je altijd een lijst met files die zijn aangepast ten opzichte van een workspace die is gekozen. Het verdient dan ook de aanbeveling altijd een workspace aan te maken per team project, zodat de lijst met uitgecheckte files altijd overzichtelijk blijft.



FIGUUR 1: WORKSPACE.

Cloaking

In een workspace geef je aan dat er een mapping is tussen een locatie in version control en een locatie op de lokale harde schijf. Zoals in Figuur 1 is te zien, zal een dergelijke mapping worden aangegeven als een active mapping. Naast een active mapping kun je ook een zogenaamde cloaked mapping aanmaken. Deze optie is bedoeld voor locaties in version control die je juist wilt uitsluiten bij het ophalen van de sources. Denk daarbij aan een mapping naar een folder die bestanden bevat die je niet dagelijks gebruikt. Een typisch voorbeeld daarvan zijn graphics, of een branch van een vorige versie van het product waar je aan werkt. Die sources gebruik je niet dagelijks, maar zijn al wel snel tientallen Megabytes groot. Het iedere keer ophalen van deze bestanden kost naast tijd ook bandbreedte terwijl je ze niet dagelijks gebruikt. Zoals eerder aangegeven is een best practice één mapping aan te maken op een root niveau of vlak daaronder in de version control boom. Door een extra mapping aan te maken en daar de optie Cloaked op aan te zetten kun je er voor zorgen dat een bepaald deel van de versioncontrolboom niet wordt opgehaald bij een commando voor het ophalen van de sources. ("get", "get Latest", "get specific version")

Shared Check-out model

Voor mensen die gebruik hebben gemaakt van Visual Source Safe, is het van zelfsprekend dat als je een bestand uitcheckt deze dan direct gelockt is voor alle andere gebruikers. Verder is het in Source Safe inchecken van de code ook een directe operatie waarna de broncode weer wordt vrijgegeven. Als je gebruik maakt van TFS, dan gaat dit iets anders. Ten eerste is het zo dat als je een bestand uitcheckt dit standaard een zogenaamde shared checkout is. Dat betekend dat het bestand niet exclusief voor jou gelockt is, maar dat je collega's hetzelfde bestand ook kunnen uitchecken. Dit betekent dat bij het inchecken het dus zo kan zijn dat een ander de code in de tussentijd ook heeft aangepast. In dat geval spreken we van een mergeconflict dat zal moeten worden opgelost. Dit wordt bij het inchecken kenbaar gemaakt door een waarschuwing dat er geen code is ingecheckt en dat je via het "pending changes" window het conflict kunt oplossen. In figuur 2 is de conflict resolution tab van het pending changes window weergegegeven.

FIGUUR 2 :CONFLICT RESOLUTION TAB

Je kunt hierin zien dat een conflict is opgetreden bij het inchecken van de file window1.xaml.cs. In dit geval is een aanpassing gemaakt in exact dezelfde methode in de source, dus daarom is de auto merge optie niet beschikbaar. Auto merge biedt de optie om de TFS server de merge voor je te laten uitvoeren, waarna je zelf de code eerst nogmaals controleert door bijvoorbeeld een compile en unit test uit te voeren, voordat je nogmaals de checkin uitvoert die daarna wel direct doorgang kan vinden. De optie "Take server", geeft aan dat je de lokale versie wilt overschrijven, de changes van je collega overneemt en je eigen aanpassingen overschrijft. "Keep local", betekent dat je de aanpassingen van je collega overschrijft met die van jezelf. De laatste optie is dat je gebruik maakt van een merge tool om de merge handmatig zelf uit te voeren. Dit kan door de standaard mergetool te gebruiken, of door een mergetool naar keuze, die je zelf kunt instellen voor dit soort conflicten.

Zoals ik al aangaf zijn standaard de checkout operaties shared. Dit gedrag kun je veranderen door deze op de server aan te passen naar exclusive check out. Doe dit niet te snel en kijk goed naar de omvang van je team en of het multiple checkout principe voor je team niet juist veel prettiger werkt. Toen ik hier zelf voor het eerst mee werd geconfronteerd, was mijn eerste reactie dat ik dit niet wilde. Echter je zult merken dat dit model een stuk prettiger werkt en dat je na een korte periode niet meer terug wilt naar het exclusive checkout model.

Pending operaties

Een andere in het oog springende verandering voor Source Safe gebruikers is dat zodra je een bestand wilt toevoegen aan version-control, dat dit resulteert in een zogenaamde "pending Add" ope-

ratie op de file. Eigenlijk geldt dit voor alle operaties die je uitvoert op version control. Als je een rename uitvoert zal dit een "pending rename" operatie worden en bij een branch een "pending branch". Een checkout zal worden geadministreerd als een "pending Edit". Al deze pending operaties geven aan dat je bij version-control hebt aangegeven wat je van plan bent te gaan doen met de betreffende bestanden. Pas op het moment dat je een complete set aan files gereed hebt ga je over tot een commit van de operaties die je hebt uitgezet. Vervolgens wordt dan door het commando Check in pending changes, de commit uitgevoerd en de files daadwerkelijk aangepast op de server.

Changesets

Als je een setje bestanden hebt aangepast en een aantal van de pending operaties wilt committen op de server, dan krijg je te maken met de check in dialog. Deze dialoog bevat een aantal zaken die van belang zijn verder te beschrijven. De dialoog is weergegeven in figuur 3.

FIGUUR 3: CHECK IN DIALOOG.

Ten eerste is dit een lijst met bestanden die worden ingecheckt op de server. Als tweede is er een lijst met Work Items die je kunt associëren met deze changes. De gedachte daar achter is dat je in versiebeheer graag wilt weten waarom een change is uitgevoerd. In TFS administreer je werk met work items. Dit kunnen taken maar ook bugs of change requests zijn. Door een checkin operatie te koppelen aan een of meerdere work items wordt hiermee de traceerbaarheid van changes mogelijk. Verder heeft men de optie zogenaamde review notes of security notes toe te voegen. Als laatste is een lijst met zogenaamde policy overtredingen beschikbaar. In TFS is het namelijk mogelijk om op de versiebeheerserver policies af te dwingen, waar de code en veranderingen aan moeten voldoen, voordat je mag inchecken. Daarbij kun je denken aan zaken als het verplichten dat iedere change die wordt ingecheckt een associatie moet hebben met een work item of dat XML documentatie aan moet staan op de project files. Er wordt een standaard set van checkin policies geleverd door Microsoft en er zijn er een aantal beschikbaar via de TFS community, maar je kunt ook zelf relatief eenvoudig een policy schrijven en afdwingen. Policies kunnen ook worden genegeerd, hiervoor dient de ontwikkelaar zijn motivatie voor op te geven en dit wordt dit op de server wel geadministreerd, zodat dit altijd later nog traceerbaar is. Verder kan op het inchecken een alert worden ingesteld, waardoor een email kan worden verstuurd op het moment dat wordt ingecheckt.

Als je nadat je alle onderdelen in de dialoog hebt doorgelopen en het "check in" commando hebt geklikt, wordt definitief de check in uitgevoerd. Hierbij wordt dan vervolgens een zogenaamde "changeset" aangemaakt. Een change set is de consistente set aan gegevens die alle informatie bevat over wat er is ingecheckt. Dit is naast de set aan bestanden, dus ook de work items die geassocieerd zijn en mogelijke check in policies die zijn gecontroleerd en eventueel zijn overtreden. Verder zit in een changeset ook informatie in over wie de checkin heeft uitgevoerd en op welk tijdstip dit is gebeurd. Een changeset is dus de eenheid van een checkin. Als je in de historie van een bestand gaat kijken, kun je zien in welke change sets dit bestand deel heeft uitgemaakt. Wat daaraan prettig is, is dat je altijd de set aan changes in relatie met elkaar kunt zien, en niet alleen de changes in het bestand. Dit maakt het geheel een stuk overzichtelijker.

Shelving

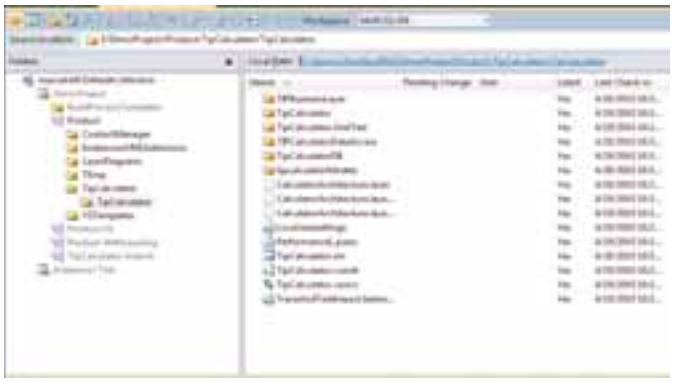
Het komt nog wel eens voor dat je een wat langere periode aan het werk bent met een feature in de software en dat je nog niet helemaal klaar bent, maar wel graag de sources wilt veilig stellen op de server. De code is op dat moment nog niet geschikt om daadwerkelijk te worden ingecheckt dus je kunt op dat moment de daadwerkelijke checkin nog niet uitvoeren. Voor deze situatie heeft TFS een extra optie die de naam "shelving" heet. Bij het shelven van de changes wordt de changeset niet aan een versioncontrol branch toegevoegd op een speciale locatie die een shelve heet. Hierbij zijn de sources wel op de server neergezet, maar zijn ze nog geen onderdeel van de branch. Shelvesets zijn tevens heel nuttig voor andere acties, zoals het laten uitvoeren van een review door een collega. In combinatie met de team build omgeving kun je een build uitvoeren met een shelveset om te zien of de build niet wordt gebroken als je deze shelveset zou inchecken. Dit is ook een mechanisme dat door Team Foundation server zelf wordt gebruikt voor het realiseren van een zogenaamde gated-checkin. Gated checkin is de optie om de aanpassingen die je wilt inchecken pas door te voeren in de branch als de shelveset kan worden gebouwd en geen fouten oplevert tijdens compilatie, unit testing, etc. Dit is dus een kwaliteitscontrole vooraf, om er voor te zorgen dat de sources die je incheckt nooit de build zouden breken.

Branching

Nu is al een aantal malen de term branch gebruikt, maar wat wordt daar nu eigenlijk mee bedoeld? Een branch is in de TFS een folder met daaronder een set aan bronbestanden. Een folder in version control is de locatie waar een set aan bronbestanden zich bevinden. Zodra er van een set aan bestanden een kopie wordt gemaakt, en daarbij de relatie met de originele bestanden moet worden geadministreerd, wordt er gesproken van een branch. Het kan bijvoorbeeld zo zijn dat je de productie versie van het systeem moet onderhouden, maar ook aan een nieuwe versie wil werken. Dus een branch is in essentie een kopie van de bronbestanden of een hele boom aan bestanden, met als doel dat de relatie tussen de kopie en het origineel te administreren. Dit wordt gedaan zodat men gescheiden aanpassingen kan maken op beide kopieën en in een later moment in de tijd de verschillen tussen

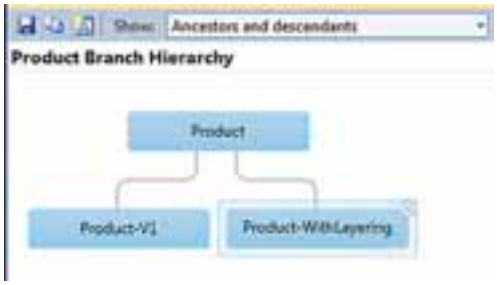
Shelvesets zijn heel nuttig voor andere acties, zoals het laten uitvoeren van een review door een collega.

beiden te kunnen bepalen om ze vervolgens mogelijk ook weer samen te voegen. Dit samenvoegen wordt een "merge" genoemt. Daarom wordt de term branch en merge vaak in samenhang met elkaar gebruikt. In figuur 4 is de versioncontrolexplorer te zien en daar is zichtbaar dat we kijken naar de Workspace Marcelv04.



FIGUUR 4: VERSION CONTROL EXPLORER.

In deze workspace zijn twee team projecten beschikbaar te weten DemoProject en Endeavour Test. Een team project is het hoofdniveau van de version control structuur waar alles voor een team project op ge-scoped wordt. Onder een team project zijn een aantal folders zichtbaar. De "Product" folder wordt gekenmerkt door een ander type symbool. Dit symbool geeft aan dat hier sprake is van een branch. Onder de "product" branch, zijn een aantal andere folders te zien. Dit zijn normale folders die onderdeel zijn van de branch. Vervolgens zie je ook dat er nog een aantal andere branches aanwezig zijn in het DemoProject team project. Dit zijn de branches "Product-V1", "Product WithLayering" en "TipCalculator Branch".
De vraag is echter wat de onderlinge relatie is tussen deze verschillende branches. Dit is in TFS te visualiseren. In Figuur 5 is de branch visualisatie weergegeven en is te zien dat alle branches een branch zijn van de product branch.



FIGUUR 5: BRANCH HIERARCHIE.

Het kiezen van een juiste branch hiërarchie voor een product en een team is een onderwerp waar een heel boek aan kan worden besteed. Er is op MSDN een speciale branching guideline beschikbaar die aangeeft wat gebruikelijke strategieën zijn om om te gaan met branching.(http://tfsbranchingguideiii.codeplex.com/) Mocht je voor de keuze staan een dergelijke strategie op te zetten voor je team, kan ik je deze guidelines van harte aanbevelen.

Changeset tracking

We hebben het al eerder gehad over changesets. Een changeset wordt ingecheckt op een branch. Als er een bug wordt oplost en deze change wordt ingecheckt, dan kun je voor de situatie komen dat je moet zien te achterhalen of een bepaalde aanpassing die is gemaakt in de software nu wel of niet beschikbaar is in een bepaalde branch. In de 2010 versie van TFS, heeft men hiervoor een extra visualisatie beschikbaar gemaakt. Het is nu mogelijk om gebruik te maken van zogenaamde changeset tracking visualisatie. In deze visualisatie wordt zichtbaar gemaakt in welke branches een bepaalde changeset is doorgevoerd. Deze is in twee varianten beschikbaar. Dit is zichtbaar in figuur 6 & 7.

FIGUUR 6: CHANGE SET TRACKING HIERARCHIES.

FIGUUR 7:CHANGE SET TRACKING IN DE TIJD.

De eerste is de hiërarchische view en de tweede is de timeline view. In de hiërarchische view, is zichtbaar dat changeset 37 momenteel alleen in de "product V1" branch is doorgevoerd, maar nog niet in de product branch. Hetzelfde is ook zichtbaar in de timeline view. Deze laatste view is zeer nuttig om te zien hoe een change set zich in de tijd door verschillende branches is doorgevoerd.

Conclusie

Met de komst van TFS 2010 is een einde gekomen aan het Visual Source Safe tijdperk. In dit artikel heb je kunnen lezen met welke nieuwe terminologie je als source safe gebruiker geconfronteerd zult worden als je de stap gaat maken naar TFS als vervanger van je bestaande versiebeheer product. Met de nieuwe concepten zoals changesets, shelvesets en de optie voor het visualiseren van branch hiërarchieën en changesets door de tijd, is het volgen van changes door de tijd en de reden waarom changes zijn doorgevoerd een stuk eenvoudiger geworden. Doordat TFS nu ook een standaard onderdeel is geworden van de MSDN subscription, is nu het gebruik van TFS ook geld technisch binnen handbereik gekomen. Mijn verwachting is dan ook, dat door deze aanpassing in de manier waarop de server beschikbaar is gekomen voor alle visual studio gebruikers, TFS de defacto versioncontrolserver zal worden van de toekomst en dat vele ontwikkelaars er met veel plezier gebruik van zullen maken.

Marcel de Vries, is Technology Manager Microsoft bij InfoSupport. Tevens is hij Regional Director en MVP van Microsoft.