Smart Clients

Een aantal jaren geleden is de smart client-applicatie ontstaan als nieuw model tussen de rich client (de Windows-applicatie) en de thin client (de webapplicatie). Het idee van de smart client is dat hij de voordelen van beide combineert en de nadelen wegneemt.

Smart Clients for dummies

Acropolis in toekomstig .NET Framework

Harco Dijkstra
Ordina

De belangrijkste eigenschappen van een smart client zijn de gebruikersvriendelijke user-interface, de eenvoudige installatie en de vele mogelijkheden van data-uitwisseling via bijvoorbeeld webservices. Bij het opstarten van de smart client kan bijvoorbeeld automatisch gecontroleerd worden of de laatste versie gebruikt wordt; indien nodig kan deze automatisch worden gedownload. Een andere mogelijkheid is dat bij het opstarten van de smart client de recentste gegevens worden gesynchroniseerd met de server. In de afgelopen jaren heeft Microsoft, via *Patterns & Practices*, een tweetal bouwstenen uitgebracht die het bouwen van een smart client eenvoudiger moeten maken. De eerste is het *Composite UI Application Block (CAB)*, dat het mogelijk maakt complexe applicaties te bouwen op basis van herbruikbare en ontkoppelde componenten. CAB gaat uit van het *Model View Controller (MVC)* ontwerppatroon of het daarvan afgeleide Model View Presenter (MVP) ontwerppatroon om de gewenste herbruikbaarheid en ontkoppeling te krijgen. Het blijkt dat veel ontwikkelaars moeite hebben met het bouwen van smart clients op basis van CAB. Dit komt deels door de vele standaardmogelijkheden die CAB biedt en deels door de vele keuzes die er gemaakt moeten worden om een smart client te ontwikkelen. De learning curve van CAB is groot en dat schrikt veel ontwikkelaars al bij voorbaat af.
Om de complexiteit van het gebruik van CAB te verminderen heeft Patterns & Practices de Smart *Client Software Factory (SCSF)* uitgebracht. Deze biedt een ontwikkelaar de mogelijkheid om veel van de handmatige werkzaamheden in CAB geautomatiseerd uit te voeren op basis van verschillende recepten die vanuit Visual Studio kunnen worden gestart. Hierbij kun je denken aan het toevoegen van een view, het publiceren van een event of het abonneren op een event. Dit maakt het leven van een ontwikkelaar een stuk eenvoudiger, maar het blijft gebaseerd op CAB, waardoor je nog steeds geconfronteerd wordt met de al eerder genoemde complexiteit als je iets anders dan de standaard functionaliteit nodig hebt.
Om de ontwikkeling van *smart clients* te vereenvoudigen is Microsoft op dit moment bezig met een nieuw project met de codenaam *Acropolis*. Microsoft heeft zich tot doel gesteld dat het definiëren, ontwikkelen, configureren, uitrollen en managen van een smart client veel eenvoudiger moet worden, dan nu met CAB en SCSF het geval is. Acropolis is een uitbreiding op het .NET Framework en bestaat uit run-time ondersteuning, design-time ondersteuning en standaard functionaliteit die door ontwikkelaars als basis gebruikt kan worden. Acropolis biedt een eenvoudige manier om herbruikbare componenten te bouwen die je kunt samenvoegen tot een smart client. Op dit moment is Acropolis beschikbaar als een *Community Technology Preview (CTP)* en werkt het alleen met het .NET Framework 3.5 en Visual Studio 2008 Beta 2.

Overzicht

Acropolis-applicaties kunnen worden gebouwd met behulp van de onderdelen zoals vermeld in tabel 1.

Term	Betekenis
Part	Basisbouwsteen van *Acropolis*, die een deel van de gewenste functionaliteit biedt. Bestaat uit twee strikt gescheiden onderdelen, te weten de businesslogica en de *PartView*.
PartView	Het user-interface-gedeelte van een Part, bij een Part kunnen verscheidene PartViews worden gedefinieerd.
Form	Een container die één of meer *Parts* bevat. De verschillende *Parts* kunnen gebruikt worden om een bepaald scenario of een gebruikerstaak mogelijk te maken. Een Form is zelf ook een *Part* en kan dus als geheel weer op een ander *Form* of in de *Shell* gebruikt worden.
Service	Basisbouwsteen van *Acropolis*, dat een deel van de gewenste functionaliteit biedt, waarbij in tegenstelling tot een *Part* geen user-interface nodig is. Een *Service* is te gebruiken op applicatie-niveau, voor bijvoorbeeld logging of autorisatie, maar kan ook functionaliteit bieden die slechts door één *Part* gebruikt wordt.
Service Dependency	Is de mogelijkheid voor een *Part, Form* of *Shell* om aan te geven van welke *Services* ze afhankelijk zijn om hun businesslogica correct uit te kunnen voeren.
Shell	Een container die de verschillende *Acropolis Parts, Forms* en *Services* samenvoegt tot een samengestelde applicatie. De *Shell* maakt het mogelijk dat de verschillende onderdelen kunnen samenwerken en heeft een aantal extra verantwoordelijkheden, zoals het instellen van de look-and-feel, de navigatiemogelijkheden en de basislay-out van de Acropolis-applicatie.
Connection Point	Het middel om, binnen een Acropolis-applicatie, de communicatie tussen de verschillende *Parts*, PartViews, *Forms*, *Services* en de *Shell* mogelijk te maken.
Navigation Controller	Bepaalt hoe de navigatie verloopt tussen de verschillende *PartViews*, waarbij het mogelijk is om dit naar wens uit te breiden of te vervangen door een ander specifiek navigatiemodel.

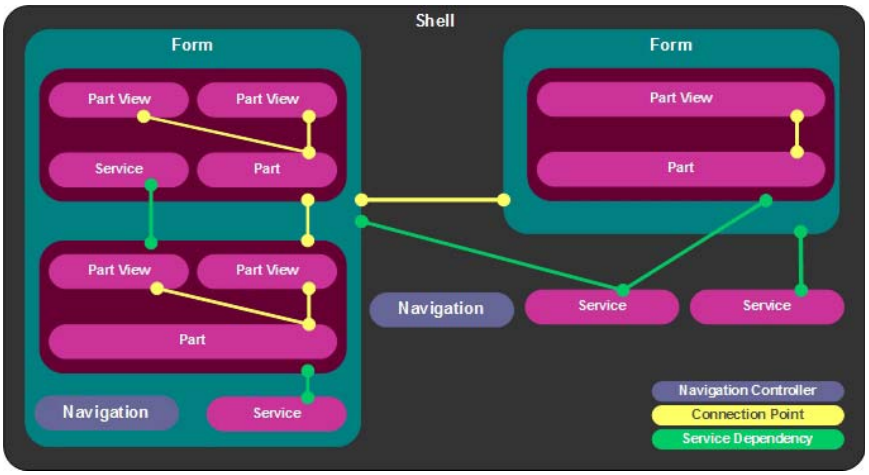
Tabel 1

De vorengenoemde Acropolis-onderdelen en hun samenhang staan in figuur 1 uitgelegd.

Part/PartView

De definitie van een Part is een ruim begrip, maar in feite is het een brok herbruikbare functionaliteit, bestaande uit een stuk businesslogica, het Part, en een of meer user-interfaces, de *PartViews*. Het is wat betreft functionaliteit meer dan een standaard user-control, maar minder dan een volledige applicatie. Een belangrijk verschil is dat bij een user-control de code van de user-interface en de code van de businesslogica nauw met elkaar verweven zijn. Om de herbruikbaarheid te bevorderen is dat juist strikt gescheiden bij Acropolis. Dat is ook de hoofdreden dat er gekozen is om het op te delen in een Part en een of meer bijbehorende PartViews. De achterliggende reden van deze scheiding is dat er op deze manier verschillende PartViews gemaakt kunnen worden die allemaal de businesslogica van het Part ontsluiten. De keuze welke PartView gebruikt wordt, kan gebaseerd worden op de rol die je binnen de Acropolis-applicatie hebt, een gebruikersvoorkeur of een andere reden. Verder bevordert deze strikte scheiding dat de koppeling tussen het Part en de PartView tot een minimum beperkt wordt. Door het definiëren van *Connection Points* wordt een interface afgesproken, tussen het Part en de PartViews. Hierdoor is het eenvoudig om nieuwe PartViews toe te voegen of te wijzigen, zolang ze maar gebruik maken van de afgesproken *Connection Points*. Deze verzameling Connection Points wordt het *Part-View Contract* genoemd, omdat het een contract is waar beide partijen zich aan dienen te houden.

Een *Part* is verantwoordelijk voor het uitvoeren van een bepaald stuk businesslogica binnen het geheel van de Acropolis-applicatie. Het is van belang dat de opdeling in verschillende Parts en bijbehorende PartViews weloverwogen wordt gemaakt. Deze opdeling bepaalt in grote mate hoe flexibel de verschillende onderdelen samengevoegd en hergebruikt kunnen worden. Een PartView is verantwoordelijk voor het tonen van de data, het correct reageren op wijzigingen door de gebruiker en het verwerken van wijzigingen die vanuit het Part komen. Door de strikte loskoppeling tussen het Part en de PartView is het mogelijk de PartViews met verschillende technieken te maken, mits ze zich aan de principes van de Connection Points conformeren. De Connection



Figuur 1 : Onderdelen Acropolis

Points zijn gebaseerd op veelgebruikte ontwerppatronen die door de meeste technieken ondersteund worden of met kleine moeite zelf gemaakt kunnen worden. De huidige CTP focust overigens op ondersteuning van op WPF gebaseerde PartViews en laat in het midden welke andere technieken ze van plan zijn te gaan ondersteunen. De interessante vragen zijn natuurlijk of ASP.NET, WinForms of SilverLight ondersteund gaan worden. Hierover zijn op de verschillende forums vragen gesteld, maar de antwoorden tot nu toe gaan niet verder dan dat ze dit in overweging nemen.

Form

Een Form, die zelf ook een Part is, kan bestaan uit een of meer Parts, met bijbehorende PartViews. Deze onderdelen werken samen om een bepaalde gebruikerstaak mogelijk te maken. Om de minimale koppeling te behouden en de herbruikbaarheid te bevorderen, bestaat een Form dus uit verwijzingen naar andere Forms of Parts. De communicatie tussen de verschillende delen verloopt via de Connection Points van de Parts en de extra Connection Points van de Form zelf. Ook kan een Form een Navigation Controller toevoegen die de navigatie tussen de Parts of Forms mogelijk maakt.

Service/Service Dependencies

Een service in Acropolis is een stuk businesslogica dat niet gekoppeld is aan een bepaalde user-interface. Een service kan worden gebruikt door verschillende Acropolis-onderdelen, zoals een Part, Form of Shell. Er moet aangegeven worden van welke services ze afhankelijk zijn. Deze afhankelijkheid wordt aangeven met een *Service Dependency* in XAML, waarvan hieronder een voorbeeld staat. Dit voorbeeld geeft aan dat er Service Dependency is met een AuthorizationService, die van het type Security.IAuthorizationService is. Deze is verplicht, omdat anders de functionaliteit niet uitgevoerd kan worden.

```
<PartFx:Part.ServiceDependencies>
  <ServiceDependency
                  Name="AuthorizationService"
                  ServiceType="{x:Type   Security.
IAuthorizationService}"
IsRequired="True"/>
</PartFx:Part.ServiceDependencies>
```

Omdat het niet gewenst is dat elk Part referenties legt naar de door hem te gebruiken services is er voor gekozen om de services via Dependency Injection beschikbaar te stellen aan het Part. De functionaliteit van een service wordt bepaald door de gedefinieerde interface, waardoor er verschillende uitvoeringen van een service gemaakt kunnen worden. Een Part heeft de mogelijkheid bepaalde services aan te bieden aan andere Parts binnen de Acropolis-applicatie. Deze services kunnen zowel in het Part worden geïmplementeerd, maar kunnen ook buiten het Part gelegd worden, mits de gespecificeerde interface geïmplementeerd wordt. De aangeboden services worden net als de *Service Dependencies* gedefinieerd in XAML. In onderstaand voorbeeld staat dat de Security.IAuthorizationService wordt aangeboden door deze Acropolis-applicatie.

```
<AcropolisApplication.Services>
<Security.IAuthorizationService/>
</AcropolisApplication.Services>
```

Shell

De Shell is de basis van een Acropolis-applicatie, waarin de verschillende Parts, Forms en Services samengevoegd worden tot een consistente en samengestelde applicatie. Ter ondersteuning wordt hierbij gebruik gemaakt van de Acropolis run-time, waarin de benodigde ondersteunende functionaliteit zit. Je moet hierbij denken aan functionaliteit die bijvoorbeeld command-routing, event-routing en life-time controllers biedt. Daarnaast definieert de Shell ook de lay-out en de navigatiemogelijkheden van de Acropolis-

Een Form, die zelf ook een Part is, kan bestaan uit een of meer Parts, met bijbehorende PartViews

Term	Betekenis
ComponentProperty	Voor het delen van een property met een *Acropolis* component.
ComponentNotification	Voor het op de hoogte brengen van een wijziging in een *Acropolis* component.
ComponentCommand	Voor het initiëren van een stuk functionaliteit in een *Acropolis* component.
ComponentData-Publisher	Voor het ontvangen van data van een *Acropolis* component.
ComponentDataUpdater	Voor het wijzigen van data van een *Acropolis* component.
ComponentDataProvider	Voor het verzenden van data aan een *Acropolis* component.

Tabel 2

Term	Betekenis
Navigation Manager	Bepaalt de logische navigatie op basis van een gedefinieerde set van regels.
Navigator	User-interface die de gebruiker de mogelijkheid geeft om via de *Navigation Manager* door de applicatie te navigeren. Bestaat over het algemeen uit user-interface-onderdelen die de navigatie mogelijk maken en onderdelen die aangeven waar de gebruikers in de applicatie is.
Part Pane	Bepaalt de lay-out van de Parts, waarbij slechts een Part tegelijk getoond kan worden.
Layout Pane	Bepaald de lay-out van de Parts, waarbij verscheidene Parts tegelijk getoond kunnen worden.

Tabel 3

Ook Connection Points worden, net als de services en service-dependencies aan een Acropolis-onderdeel toegevoegd via XAML

applicatie. Op dit moment kan er wat betreft navigatie gekozen worden voor de mogelijkheid om slechts één Part tegelijk te laten zien of meer Parts tegelijkertijd. Onthoud dat een Form ook een Part is en dat dit navigatiemodel minder beperkt is dan het in eerste instantie lijkt. Als er gekozen wordt voor meer tegelijk, kan gekozen worden uit een lay-out gebaseerd op tabs, een lay-out gebaseerd op splitters en een lay-out gebaseerd op vrij te positioneren schermen. Als deze standaard geboden lay-outs niet voldoen, is het mogelijk een eigen afgeleide te maken die wel de gewenste lay-out biedt.
Verder biedt de Shell nog de mogelijkheid om aan te geven welk thema er gebruikt wordt. Een thema bepaald het uiterlijk van de Acropolis-applicatie en ontwikkelaars hebben de mogelijkheid hun eigen thema's toe te voegen. Ondersteuning van verschillende thema's is iets dat tot op heden niet op eenvoudige wijze gemaakt kon worden, op het instellen van wat kleuren en het font na. Door de standaard ondersteuning van WPF zijn de mogelijkheden hiervoor veel uitgebreider geworden. Ook kan in de Shell worden aangegeven hoe de overgang tussen verschillende PartViews verloopt, waarbij de uitgebreide mogelijkheden van WPF gebruikt kunnen worden.

Connection Point

Om ontkoppeling tussen de verschillende Acropolis-onderdelen mogelijk te maken, worden interacties hiertussen gedefinieerd als Connection Points. Deze bieden indirecte communicatie tussen Parts, PartViews, Forms en Services. De ontwikkelaar definieert de Connection Points die nodig zijn om de gewenste functionaliteit te implementeren.
Acropolis biedt op dit moment zes verschillende Connection Points, maar het is mogelijk om eigen specifieke types toe te voegen. Gezien de algemene opzet van de standaard Connection Points zal dit slechts in een beperkt aantal situaties noodzakelijk zijn.
Ook Connection Points worden, net als de services en service-dependencies aan een Acropolis-onderdeel toegevoegd via XAML, waarbij de naam en de zichtbaarheid ten opzichte van andere Acropolis-componenten opgegeven dienen te worden; en een aantal additionele parameters afhankelijk van het type Connection Point. In onderstaand voorbeeld worden twee Connection Points gespecificeerd, te weten een ComponentProperty, met de naam PartInfo en zichtbaar voor andere Parts en PartViews. Daarna wordt een ComponentCommand gedefinieerd met de naam SaveFileCommand, waarbij aangegeven wordt welk standaard commando door dit Connection Point wordt afgehandeld. Verder is dit Connection Point alleen maar zichtbaar voor andere Parts, en wordt bij uitvoering van dit commando een functie aangeroepen met de naam SaveCommand_CommandExecuted.

```
<AcropolisComponent.ConnectionPoints>
  <!-- Property Connection Point -->
<ComponentProperty
Name="PartInfo"
Visibility="Part; View"/>
  <!-- Command Connection Point -->
<ComponentCommand
Name="SaveFileCommand"
Handles="ApplicationCommands.Save"
Visibility=Part
CommandExecuted="SaveCommand_CommandExecuted"/>
</AcropolisComponent.ConnectionPoints>
```

Navigation Controller

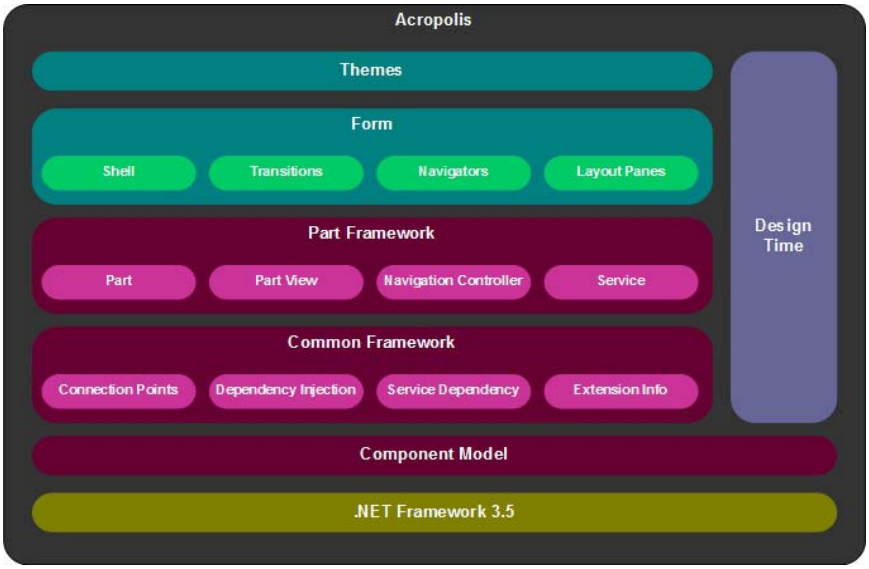
Navigatie binnen een applicatie is altijd een belangrijk onderdeel, omdat het in grote mate het gebruiksgemak bepaalt. Een goede navigatie maakt het voor een gebruiker eenvoudig de gewenste taak zonder verstoringen of onlogische stappen uit te voeren. Acropolis onderkent dit en maakt het eenvoudig om navigatie aan jouw Acropolis-applicatie toe te voegen, en er voor te zorgen dat de navigatie tussen verschillende Parts eenduidig verloopt. Acropolis wordt geleverd met een aantal standaard Navigation Controllers

en indien nodig kunnen specifieke navigatiestructuren worden toegevoegd door het implementeren van zelfgemaakte navigation controllers. Navigation controllers kunnen worden toegevoegd op de Shell en op de verschillende Forms, waardoor de mogelijkheid bestaat om een ander specifiek navigatiemodel te gebruiken in een Form.

Een navigation controller is opgebouwd uit de onderdelen vermeld in tabel 3 op pagina 20.

Details

De lagenstructuur van Acropolis is in het volgende figuur weergegeven, waarin de tot nu toe behandelde onderdelen alle naar voren komen, zoals het Part, de PartView, Form, Shell, Service, Navigation Controller, en Connection Points.



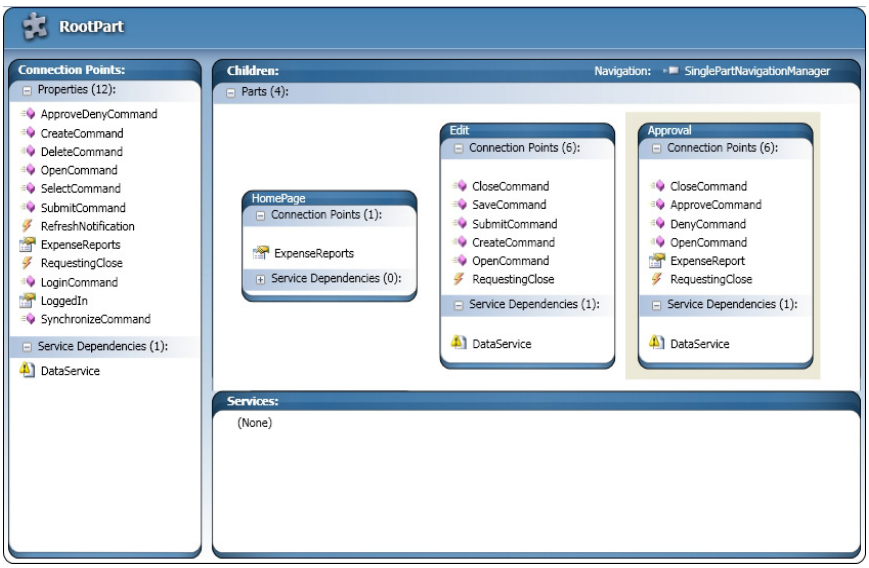
Figuur 2: Acropolis-lagen

Om het ontwikkelen van Acropolis-applicaties eenvoudig te maken, heeft men ervoor gekozen om de XAML ook te visualiseren in een designer waarmee eenvoudig de structuur van de Shell, een Form of een Part bekeken en gewijzigd kan worden. In figuur 3 staat een voorbeeld van hoe een Form in Acropolis er uitziet in de designer. Hierin staan aan de linkerkant de beschikbare Connection Points en Service Dependencies gedefinieerd en rechts staat een aantal *Parts*. Daaronder worden de Services getoond die door dit Part worden aangeboden, maar daar wordt in dit voorbeeld geen gebruik van gemaakt. De verschillende Parts hebben hun eigen Connection Points en Service Dependencies. Helemaal rechtsboven staat aangegeven welk navigatiemodel gebruikt wordt in dit Form. Dat houdt in dit geval in dat te allen tijde slechts een van de Parts actief kan zijn.

Geen nieuwe CTP's

Wat betreft concept kan gesteld worden dat Acropolis een veelbelovende techniek is, die in tegenstelling tot CAB en SCSF een minder grote leercurve zal hebben en de implementatie van samengestelde applicaties eenvoudiger zal maken. Het is de bedoeling dat als Acropolis gereleased wordt het op zijn minst dezelfde mogelijkheden zal ondersteunen als CAB en SCSF op dit moment. De huidige CTP is nog lang niet op dat niveau, maar een groot deel van de benodigde ondersteunende functionaliteit zit er al wel in. Acropolis introduceert weinig baanbrekende of vernieuwende concepten, maar de kracht zit in de combinatie van de bestaande concepten en de ondersteuning hiervan door de Acropolis run-time. Een ontwikkelaar hoeft zich niet of in elk geval minder bezig te houden met het koppelen van de losse onderdelen van de applicatie en zich kan focussen op de gewenste businesslogica. De scheiding tussen Parts en PartViews en de flexibele manier van het gebruik van services maakt het mogelijk om verschillende werkzaamheden eenvoudiger te verdelen tussen teamleden, waarbij ieder zich kan focussen op zijn of haar specialisme. Verder wordt door deze scheiding de testbaarheid van de losse componenten vergroot en is de onderhoudbaarheid van de applicatie hoger. Het geheel bestaat uit verschillende afgebakende onderdelen die een lage koppeling hebben, waardoor delen eenvoudig zijn uit te wisselen of aan te passen, zonder dat dit ingrijpende gevolgen heeft voor andere delen van de Acropolis-applicatie. Door deze ontkoppeling zijn ook de mogelijkheden tot hergebruik tussen projectteams een stuk groter geworden.

Het is overigens op dit moment erg onduidelijk wanneer Acropolis de status van een officiële release zal bereiken en welke functionaliteit er dan wel of niet in zal zitten. Het project zit op dit moment in een vroeg stadium en dit zal in de



Figuur 3 : Part Designer

loop van de tijd wel duidelijker worden. Op dit moment vraagt Microsoft aan gebruikers om hun mening te geven en hun wensen kenbaar te maken, zodat ze zich bezig kunnen houden met de onderdelen die door de gebruikers belangrijk worden gevonden. Mijn verwachting is dat er nog een aantal CTP's zullen volgen voordat een officiele release uitgebracht wordt. Met de feedback van gebruikers zal de huidige versie verder uitgebreid en geoptimaliseerd worden, waarbij breaking changes niet uitgesloten zijn. Dit betekent dus dat Acropolis op dit moment te prematuur is om in productieomgevingen te gebruiken, maar dat is gebruikelijk met CTP's. Eind oktober heeft Microsoft overigens besloten te stoppen met het Acropolis-project en om geen nieuwe CTP's meer uit te brengen. De achterliggende reden van dit besluit is dat men van mening is dat het niet een apart product moet zijn, maar dat de concepten en ideeën verwerkt moeten worden in een toekomstige versie van het .NET Framework.

Links

http://www.windowsclient.net/Acropolis/ - Downloads, Video's, Presentaties, Voorbeelden.

http://blogs.msdn.com/Acropolis/ - Blog van het Acropolis ontwikkelteam.

http://blogs.msdn.com/KathyKam/ - Blog van een van de projectmanagers van Acropolis.

http://blogs.msdn.com/DPhill/ - Blog van een van de projectmanagers van Acropolis.

http://blogs.msdn.com/RickyT/ - Blog van een van de ontwikkelaars van Acropolis.

http://blogs.msdn.com/DDietric/ - Blog van een van de ontwikkelaars van Acropolis.

http://blogs.msdn.com/BradA/ - Blog van een van de projectmanager van Acropolis.

Patches Patches Patches Patches Patches Patches Patches P

Artikelen over onderwerpen als software-ontwikkeling, Java, UML, eXteme Programming en nog veel meer vindt u in het Online Archief van Array Publications. Vaktijdschriften als Software Release, Java Magazine, Database Magazine en ons Oracle vakblad Optimize hebben hun artikelenarchief online gezet. Dankzij de heldere zoekstructuur vindt u snel wat u zoekt op www.release.nl.

Servoy verdubbelt omzet voor het vijfde opeenvolgende jaar

Servoy B.V. heeft in 2007 vergeleken met 2006 een omzetgroei van 140% gerealiseerd. Wereldwijd werken meer dan 200.000 eindgebruikers met software toepassingen die zijn ontwikkeld in Servoy. Servoy B.V. boekt sinds haar oprichting een substantieel deel van haar omzet in het buitenland, met name de USA, maar groeit nu ook fors in Nederland na de introductie van het Servoy ISV Assurance Program.

Servoy 4.0

Sevoy zal in het tweede kwartaal van 2008 haar nieuwe 4.0-versie introduceren. De belangrijkste wijziging hierin is de switch naar Eclipse als ontwikkelplatform. Jan Blok CTO van Servoy: "toen wij 7 jaar geleden begonnen met de ontwikkeling van Servoy waren er geen IDE's gebaseerd op open standaarden en moesten wij een eigen IDE ontwikkelen. Vandaag de dag is Eclipse zo populair en uitgebreid, dat het een logische keuze is om over te stappen op Eclipse. De ontwikkelaars die in Servoy programmeren profiteren zo van de krachtige editors en tools die het Eclipse platform biedt. Eclipse wordt op dit moment vier keer zoveel gebruikt als Visual Studio van Microsoft."

OutSystems telt 10.000 downloads van OutSystems Express Edition

Gratis ontwikkelomgeving voor bedrijfsapplicaties groeit in populariteit. OutSystems, leverancier van applicatieplatformen, kondigt aan dat inmiddels 10.000 gebruikers de OutSystems Express Edition hebben gedownload van de website. Met dit platform kunnen gebruikers eenvoudig applicaties op maat ontwikkelen in .NET en Java. Deze gratis versie, met een limiet van vijf gebruikers, biedt organisaties de mogelijkheid om zonder investering kennis te maken met het platform. In Nederland is de software inmiddels ruim 1.000 keer gedownload.

Het OutSystems-applicatieplatform zorgt ervoor dat ontwikkelaars en IT-professionals eenvoudig applicaties kunnen ontwikkelen, de informatie snel via het web kunnen ontsluiten en op elk gewenst moment kunnen aanpassen, zonder downtime van het systeem. Zo kunnen bedrijven die te maken hebben met veel veranderingen in hun omgeving hier eenvoudig op inspelen. Daarnaast kunnen organisaties die de volledige editie van het OutSystems-platform hebben aangeschaft, gelijk overgaan tot het terugverdienen hiervan. Dit is mogelijk omdat de kosten voor het beheer en onderhoud erg laag zijn; er is amper beheer nodig, de applicaties zijn gelijk inzetbaar, er zijn geen verborgen kosten en ook is het mogelijk om ter plekke wijzigingen door te voeren.

Community

OutSystems biedt zijn producten aan via de OutSystems community. Deze online omgeving biedt de gebruikers gratis downloads met de bijbehorende handleidingen. Ook kunnen ontwikkelaars eenvoudig kennis en ervaringen delen met en contact leggen met de OutSystems helpdesk.

Paulo Rosado, CEO binnen OutSystems: "Het succes van onze online community en de groeiende vraag naar onze gratis ontwikkelapplicatie bevestigen onze visie dat bedrijven behoefte hebben aan een innovatieve en eenvoudige oplossing. IT-afdelingen kunnen met het OutSystems-platform snel maatwerkapplicaties ontwikkelen en daarmee inspelen op veranderingen in de markt. Daardoor zijn zij altijd een stap voor op hun concurrenten."

Bedrijven als Bosch, Bristol-Myers Squibb, E.ON, Oxxio, Unilever, Van Ameyde Groep en Wasco gebruiken de OutSystems software voor het ontwikkelen van webapplicaties, mobiele toepassingen en complete bedrijfsapplicaties.