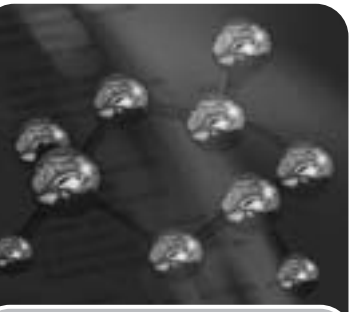

thema

Waarom een artikel over GUI's of presentatielaag oplossingen in een editie gericht op de Service Oriented Architecture? Het antwoord is eenvoudig: de presentatielaag is datgene wat onze eindgebruikers scheidt van de Service Oriented Architecture. De eindgebruikers worden er in principe niets wijzer van. Juist daarom zijn betere front-ends terdege belangrijk. Als de productiviteit en effectiviteit van onze applicaties niet verbeteren wordt het potentieel van zelfs de beste architectuur teniet gedaan.

Rich Internet Applications als service consumers

Waarom pure HTML niet voldoet

Een Rich Internet Application is het front-end van een webapplicatie welke een betere en productievere gebruikerservaring biedt dan een typische HTML interface maar wel de voordelen van een pure HTML webapplicatie heeft, zoals onder meer de beheersbaarheid en lagere uitrolkosten. Een Rich Internet Application (RIA) biedt een betere gebruikerservaring omdat het de tekortkomingen van HTML applicaties adresseert. Er zijn twee tekortkomingen van HTML user interfaces die in het oog springen.

GEBRUIKERS-INTERACTIE HTML is van oorsprong bedoeld voor het publiceren van statische, tekstgeoriënteerde documenten met beperkte gebruikersinteractie. HTML applicaties zijn opgebouwd met de welbekende bouwstenen; de hyperlink, het invoer veld, het grote invoer veld, de groep radio knoppen, de check box, het pull-down menu, de button en het plaatje.

Hoewel er andere *user interface* onderdelen lijken te bestaan (het tabblad, de kalender, het aanpasbare venster, de tooltip) worden deze "gefingeerd". Het betreft dan vaak een nieuw verzoek aan de server of een client-side oplossing met gebruik van DHTML (XHMTL, CSS en JavaScript). Met behulp van de pure HTML elementen worden deze *user interface* componenten dan gesimuleerd. Dit is een tijdrovende en complexe manier van interfaces maken, en werkt vaak niet helemaal goed.

Het ontbreekt zulke HTML oplossingen aan belangrijke mogelijkheden waar zelfs eenvoudige client/server applicaties over beschikken. Basale GUI onderdelen zoals modale vensters, directe manipulatie en feedback ontbreken. Zonder dit soort componenten is het realiseren van ergonomische en gemakkelijk te gebruiken applicaties lastig, getuige de vele slecht bruikbare webapplicaties. Het ontbreken van het soort componenten welke het uitvoeren van typische werkzaamheden eenvoudig en snel maken hebben een negatieve invloed op de productiviteit.

TEKORTSCHIETENDE PRESTATIES Het web is gebaseerd op een statusloos vraag/antwoord model. Het resultaat hiervan is dat browsers na iedere gebruikersinteractie een nieuw document moeten aanroepen en het resultaat moeten laden. Gebruikers van webapplicaties werken hierdoor met het wel bekende "click, load, refresh" verschijnsel.

Omdat complexe taken vaak worden opgedeeld in een aantal stappen maakt dat deze aanpak voor dit soort functionaliteit veel "ritjes" naar de server en terug nodig zijn. Dit resulteert opnieuw in verminderde productiviteit, een toenemende belasting van de server en een onnodige hoge netwerk overhead. Daarnaast moet er in de architectuur van de pure HTML webapplicatie voorzieningen worden getroffen om de zogenaamde *conversational state* te kunnen managen. Een standaard als Struts is hierdoor zo ingeburgerd dat het voor sommige ontwikkelaars niet eens meer duidelijk is welke tekortkoming ze in feite oplossen, maar een doel op zich worden.

Voor het uitvoeren van relatief minder vaak voorkomende taken, zoals Internet bankieren of een boek bestellen, zijn deze veelvuldige request-response cycli minder erg. Het is echter een ander verhaal, wanneer

bedrijfsapplicaties op dezelfde manier gaan werken omdat ze relatief veel complexe taken bevatten. Aan de ene kant eisen bedrijven hogere productiviteit van hun medewerkers, terwijl aan de andere kant applicaties worden geïmplementeerd waarmee men minder productief is.

Met de brede acceptatie van service georiënteerde architectuur, en het relatief gemakkelijke ontsluiten van systemen via (web)-services, is het logisch dat bedrijven kiezen voor het beschikbaar stellen van systemen via webapplicaties. Hoewel hiermee kostenbesparingen mogelijk zijn en bedrijfsprocessen gemakkelijker kunnen meebewegen met de markt, zit er ook een negatief aspect aan, dat bedrijven zich vaak niet realiseren. De productiviteit van medewerkers zal - zonder extra maatregelen - sterk afnemen, en alsof dat niet genoeg is, de werkvreugde hoogstwaarschijnlijk ook. Zeg nou zelf, zou u het fijn vinden om afgerekend te worden op het aantal gerealiseerde transacties, waarbij de applicatie u na ieder invoerscherm laat wachten totdat de nieuwe interface geladen is.

REKENKRACHT De oplossing voor deze problematiek ligt eigenlijk voor de hand. Maak gebruik van de meer dan voldoende aanwezige client-side rekenkracht en de kracht van Service Oriented Architecture door middel van een Rich Internet Application. De twee belangrijkste tekortkomingen worden hiermee adequaat opgelost. Doordat een RIA een uitgebreide set *widgets* heeft is de eindgebruiker beter in staat zijn data te manipuleren en zijn de interacties intuïtiever te maken. Zo zijn er widgets voor bijvoorbeeld datum selectie, hiërarchische structuren, interactieve tabellen, schuifjes en navigators (tabs, accordeons, stacks). Allemaal middelen waarmee rijkere gebruikersinteractie mogelijk wordt.

De tweede in het oog springende tekortkoming van HTML wordt verbeterd doordat RIA's een directe interactie heeft met de gebruiker. De gegevensuitwisseling met de server vindt letterlijk achter de schermen plaats. Daarnaast worden invoervelden veel gebruiksvriendelijker en directer gecontroleerd. Omdat de data lokaal verwerkt worden, zijn er minder interactie momenten met de server nodig. Eindeloze request-response cycli zijn niet meer nodig.

Er zijn al aardig wat oplossingen beschikbaar waarmee bedrijven aan de slag kunnen om de voordelen van Service Oriented Architecture en de beheersbaarheid van webapplicaties voor zich te laten werken, waarbij de gebruikerservaring en productiviteit niet hoeven in te boeten. RIA's maken het mogelijk om moderne user interface principes in tandem te laten werken met services van de multi-tier architectuur.

SOORTEN EN MATEN Wat is nu precies een Rich Internet Application? Een Rich Internet Applicatie is een applicatie welke gedistribueerd wordt via Internet technologie, welke (web)services aanroept om zich te voorzien van data en welke beschikt over een rijke set aan interface componenten waarmee de gebruikerservaring sterk verbeterd kan worden ten opzichte van losse HTML pagina's.

Er is een onderverdeling te maken voor RIA's op basis van het runtime platform. Er zijn RIA's welke met een eigen plug-in werken, die op basis van Java werken en die op basis van bepaalde browserversies werken.

- RIA's op basis van een eigen plug-in hebben het voordeel dat de plug-in veel van de specifieke code bevat waardoor ze erg compact zijn. Daarnaast biedt een plug-in de zekerheid van een heldere definitie van de runtime omgeving, ongeacht het device of het platform. Voorbeelden van RIA platforms in deze categorie zijn Macromedia Flex en Flash, Curl, DreamFactory en Laszlo systems. Deze laatste maakt ook gebruik van de Flash plug-in. Het voordeel hiervan is dat de benodigde runtime op bijna alle browsers aanwezig is, afhankelijk van de versie ongeveer 98% van alle desktops en laptops, en de plug-in is erg klein is (500k).
- Er zijn ook Rich Internet Applicaties die op basis van het Java platform werken. In plaats van een eigen plug-in maken deze gebruik van het JVM binnen de browser. Hoewel Java op veel computers geïnstalleerd is, heeft het een beperkte dekkingsgraad. Altio en NexaWeb en zogenaamde thinlets zijn voorbeelden van Rich Internet Applicaties die gebruik maken van het Java Virtual Machine.
- De laatste categorie RIA's zijn die platforms die werken met een bepaalde browser. Omdat browsers vaak niet over een XML parser beschikken zijn ze afhankelijk van een serverside component voor het feitelijke consumeren van services. Om de interface niet in zijn geheel te verversen tijdens een interactie met een webservice vallen ze vaak terug op "trucjes" zoals hidden frames of onzichtbare Flash of Java componenten. Daarnaast blijven oplossingen in deze categorie gebonden aan de mogelijkheden van HTML in de browser voor het renderen van interfaces, hierdoor voelen sommige niet zo rijk aan als andere RIA

> Met de widgets van een RIA kan de eindgebruiker zijn data beter manipuleren en de interacties intuïtiever maken

alternatieven. Voorbeelden van RIA oplossingen in deze categorie zijn; General Interface, JackBe, SCO WebFace (vultus), en last but not least ons eigen Nederlandse Backbase. Het grote voordeel van al deze platforms is dat ze niets anders nodig hebben naast de specifieke browser.

- XUL is ook een aardig RIA platform, en past niet zo goed in één van deze categorieën. Het is een open source initiatief en is oorspronkelijk bedoeld voor het Mozilla platform. Er zijn echter ook al XUL rendering engines geschreven in Flash en in Java (thinlets) zodat XUL RIA's ook in andere browsers kunnen werken.

RIA en JSF Hoe verhoudt het RIA concept zich tot de Java Server Faces definitie? Het Java Server Faces raamwerk lost voor een deel het probleem van traditionele pure HTML applicaties op door de relaties tussen verschillende Interface onderdelen (de status van de interface) op de server te beheren. Zonder gebruik van uitgebreide en platform afhankelijke DHTML is het gebruiksgemak van JSF applicaties niet beter dan pure HTML applicaties. Geavanceerde JSF applicaties waarbij intensief gebruik wordt gemaakt van DHTML bieden mogelijkheden welke lastig te realiseren zijn met alleen pure HTML en een MVC als Struts. Een goed voorbeeld hiervan is de webinterface van Documentum, webtop.

> Een plug-in biedt de zekerheid van een heldere definitie van de runtime omgeving, ongeacht het device of het platform

Veel RIA platforms zijn complementair aan server side standaarden. Het is bijvoorbeeld mogelijk een JSF naar Flex MXML rendering te maken. Struts kan nog steeds de dialoog managen met een Flex front-end en Flex is ook goed te integreren met de portlet standaard.

RIA en XForms XForms is een specificatie voor een client die de gebruiker in staat stelt XML-documenten te bewerken. De interactiviteit van de webapplicatie zit hem dan in het manipuleren van deze XML documenten, waarna ze naar de server gestuurd worden. De XForms-standaard specificeert niet alleen de user interface en de relatie met het achterliggende gegevensmodel, maar ook het proces van een XForms-applicatie, een event model en een aantal functies als aanvulling op XPath. In zekere zin heeft het W3C met de XForms definitie een functioneel ontwerp gemaakt voor een XForms RIA Client. Als de populaire browsers alle benodigde technologie bevatten waar XForms afhankelijk van is, zijn het platforms geworden waarmee RIA's gemakkelijker geïmplementeerd kunnen worden. Tot die tijd kunnen XForms RIA's gemaakt worden met de FormsPlayer plug-in.

RIA en Microsoft Het blijven toevoegen van functionaliteit aan het Document Object Model van Internet Explorer leek voor Microsoft enige tijd de oplossing voor de tekortkomingen van HTML te zijn maar met de introductie van de aan Longhorn gerelateerde technieken lijkt het erop dat er een andere weg wordt ingeslagen. Plotseling verschijnen er geen nieuwe versies meer van Internet Explorer terwijl rivaliserende browsers zoals Fire-Fox en Opera wel functionaliteit toevoegen waar eindgebruikers erg tevreden over lijken te zijn (zoals tabbed browsing). Eigenlijk is Longhorn een geavanceerd en uitgebreid RIA platform. Via de Extensible Application Markup Language (XAML) zal het mogelijk zijn in XML user interfaces te specificeren. Met deze technologie biedt Microsoft straks een hoogwaardige user interface voor zowel webapplicaties als desktop clients. Tezijnertijd is het een ideaal platform voor het consumeren van de services uit de Service Oriented Architecture.

Er zijn geruchten dat verdere uitbreiding van Internet Explorer het succes van Longhorn wel eens zou kunnen ondermijnen. De redenering is dat - wanneer Internet Explorer *de facto* de service consumer wordt - niemand nog warm zal lopen voor een geheel nieuw platform waarvoor stevig geïnvesteerd zal moeten worden in licenties… In die zin zijn de initiatieven van bedrijven als Backbase en Macromedia wellicht een doorn in het oog van Microsoft.

Dat RIA's hot zijn blijkt wel uit de volgende citaten:

"*Companies that are in the business of providing web-based applications should seriously consider rich Internet application solutions in order to move aggressively beyond the pagebased approach to improve the usability of their products and services.*" -- Matthew Berk, Senior Analyst Jupiter Research, Juni 2003.

Ook Gartner doet zijn zegje over RIA, in het licht van online retailing.

"*By utilizing rich Internet applications, companies will be able to offer users a more sophisticated online experience, and in turn increase customer retention rates and help drive new revenue streams.*" -- Mark Driver, Research Director van Gartner, Juni 2003.

In mijn ogen zijn RIA platforms een zeer welkome techniek waarmee we de effectiviteit van onze software op een hoger plan kunnen tillen.

Ervaringen met Flex

Er zijn een aantal zaken waar een goede RIA oplossing in mijn ogen aan moet voldoen, in principe zijn deze eisen niet anders dan voor andere ICT oplossingen: open standaarden, uitbreidbaarheid, goede tool-support, robuustheid, onderhoudbaarheid en component based development. Wat met name voor Rich Internet Application oplossingen geld is uitgebreide ondersteuning voor services, een goed event-model en een rijke en uitbreidbare set van user interface componenten. Voor het consumeren van web services is het noodzakelijk dat er meerdere mogelijkheden zijn voor het aanroepen: synchroon, a-synchroon, statische (of restful services), en ouderwets via http://get/post requests. Er zijn kantekeningen te plaatsen over de snelheid en overhead van het verpakken van ieder request en respons in XML berichten, dus ondersteuning voor binaire protocollen is ook een pré.

Macromedia Flex voldoet aan veel van deze voorwaarden. Door gebruik te maken van de alomtegenwoordige Flash plug-in, een robuuste ondersteuning voor een veelvoud van webservices te bieden en over een rijke en uitbreidbare verzameling van user interface componenten te beschikken is het een goede oplossing.

Puurheid Flex biedt een ontwikkelmodel dat J2EE en .NET ontwikkelaars op het lijf is geschreven; door middel van een declaratoire XML taal worden interfaces gedefinieerd. Deze worden door een server worden omgezet in de uiteindelijke user interface welke op de client tot uitvoer komen.

Flex is een standaard J2EE-component, die ondersteund wordt op alle vooraanstaande applicatie servers (ik heb het zelf getest op BEA WebLogic en Tomcat). Naast SOAP is het mogelijk om gebruik te maken van het AMF protocol om vanuit de client services op de server aan te roepen. Dit binaire protocol van Macromedia maakt het mogelijk om direct een Java component (Pojo's of EJB's via het service façade pattern.) aan te roepen waarbij Flex zorg draagt voor het omzetten van Java objecten naar Action-Script objecten en vice versa.

Action-Script is de op ECMA script gebaseerde programmeertaal waarmee het gedrag van Flex RIA's geprogrammeerd wordt. Deze taal is uitgegroeid tot een volwassen objectgeoriënteerde strongly-typed programmeertaal. Dit alles is ook IBM niet ontgaan en daarom wordt er hard gewerkt om Flex een plekje te geven in de Websphere producten, op dit moment als plug-in voor WSAD studio en Eclipse. Flex applicaties onderscheiden zich in architectonische puurheid, niet alleen door de client-tier ook daadwerkelijk naar de client te halen, maar ook door binnen deze tier goede ondersteuning te bieden voor de bekende design-patterns en best-practices:

- Duidelijke scheiding tussen informatie, opmaak en gedrag
- Object-Oriented
- Uitbreidbaarheid door middel van Component Based development
- Declaratieve interface definitie
- Design-patterns als "front-controller", "value objects", "model-view-controller" en "delegate".

De onderdelen waar Flex-applicaties mee gebouwd worden zijn de XML tags en Action-Script classes die tijdens het compileren omgezet worden in Flash bytecode. Het behoort inmiddels tot de praktijk om deze componenten uit te breiden door overerving en compositie van bestaande componenten. Bestaande Flex-componenten (XML definities en Action-Script classes) worden dan gecombineerd en uitgebreid tot eigen componenten met hun eigen namespace. Een aardig voorbeeld is de slider component. In de eerste versie van Flex was dit component toegevoegd om de uitbreidbaarheid te demonstreren. In de volgende versie van Flex heeft de slider component zijn plaats tussen de andere standaard componenten verworven.

Nieuwe tijd Het belangrijkste onderdeel van een RIA is het kunnen consumeren van (web)services. Flex RIA's roepen functionaliteit aan en consumeren de resulterende informatie in de vorm van willekeurige XML berichten door zelf een SOAP request op te bouwen en de response te verwerken (of door het oproepen van XML documenten met een *restful webservice*). De Flash plug-in biedt de hiervoor benodigde XML parser. Hierdoor kunnen services aangesproken worden zonder dat de gebruiker hiervan op de hoogte is. De bediening van de applicatie kan gewoon doorgaan, of juist geblokkeerd worden, afhankelijk van de situatie.

Daarnaast is het interessant om te constateren dat de Flash plug-in ondersteuning heeft voor socket communicatie. Hiermee wordt het mogelijk om de Flex client via een server push op de hoogte te stellen van veranderende omstandigheden, waardoor de client nieuwe requests kan uitvoeren zodat de verouderde data vervangen wordt door verse, correcte informatie. Dit biedt zoveel nieuwe mogelijkheden dat veel van onze ingebakken ideeën over internet applicaties herzien moeten worden, en dat er een nieuwe tijd aanbreekt met ongekende mogelijkheden.

Ilya Devèrs is werkzaam als senior consultant bij Capgemini. Hij houdt zich bezig met J2EE Software Architectuur, Front-end Engineering, Usability en User Centered Design.
Met dank aan Bas Peters voor de XForms informatie.