Chirojeugd-Vlaanderen vzw

WERKGROEP GAP		
13 januari 2012	**Not.**	

Aanwezig: Ben Bridts, Johan Vervloet
Verontschuldigd: Mathias Keustermans

We bespraken op Kipdorp de mogelijkheden van een API voor GAP

Wat verwachten we van een API?

In eerste instantie niet veel:

- De mogelijkheid om een lijst op te halen van de leden per afdeling
- De mogelijkheid om individuele gegevens van een lid op te halen

Sowieso bieden we in eerste instantie ook enkel read-only functionaliteit.

Proof of concept OData

We bekeken een klein projectje dat met behulp van WCF data services toegang geeft tot een in-memory (fake) ledenlijst. Hoe het werkt, staat omstandig uitgelegd op de wiki: https://develop.chiro.be/trac/cg2/wiki/API

Authenticatie

De vraag is nu: hoe laten we een client authenticeren tegenover een API?

Basic HTTP authentication

Je zou de API kunnen hosten op een webserver die basic HTTP authentication gebruikt. Voordeel is dat die webserver dan via impersonation de GAP-backend kan aanspreken. Maar het grote nadeel is dat de verleiding erg groot is om username en wachtwoord in een Javascriptje te stoppen, om gemakkelijk JQuery-calls uit te voeren op de API. Dat moet vermeden worden.

OAuth

OData wordt nogal vaak gebruikt in combinatie met OAuth. OAuth is het authenticatiemodel dat bijvoorbeeld Google en Twitter aanbieden voor hun API's. Het is een systeem dat via signing werkt, en een applicatie toelaat om de API te gebruiken als een bepaalde gebruiker, zonder diens wachtwoord te kennen.

Federated Login

Hier komt het erop neer dat je credentials die je hebt voor toepassing B gebruikt om aan te melden voor toepassing A. OpenID is een voorbeeld, maar ook SAML (dat al vaak besproken werd in de context van 'hoe kunnen we logins voor GAP hergebruiken voor de website). Ik ben niet zeker in welke mate dat bruikbaar is voor een REST-API.

WCF Data Services: de magie en IQueryable

Het voorbeeldprojectje leverde gegevens op uit een lijst met 5 leden. Die lijst zat altijd in-memory, en iedere keer de service aangesproken werd, werd die lijst gequery't.
In praktijk zitten de leden in een database, en zijn het er 100000. Ze allemaal op voorhand ophalen, is geen optie. Bovendien zijn de opvraagbare leden verschillend van user tot user.
In de standaardvoorbeeldjes van WCF Data Services werkt de data service rechtsreeks op een objectcontext van entity framework. Op die manier worden bij het evalueren van de query enkel de noodzakelijke query's naar de database gestuurd (ipv eerst heel de tabel op te vragen). Rechtstreeks op de entity's werken is voor ons niet interessant, om deze redenen:

- We zouden graag de servicelaag aanspreken om de gegevens op te vragen, zodat we onze data contracts kunnen hergebruiken voor de API
- Rechtstreeks op entity framework werken breekt onze architectuur
- De tabelstructuren die we gebruiken, zijn te ingewikkeld voor de users van de API. Dat zou eventueel wel opgevangen kunnen worden door de services op views te laten werken.

- De security moet gewaarborgd blijven. Dit zou eventueel ook kunnen op basis van views, gegeven dat de database de user kent die de query uitvoert.
- Het zou alleszins niet proper zijn.

Het beste zou zijn dat we zelf een implementatie maken van IQueryable, die werkt op onze webservices. Er zijn voorbeelden te vinden op het internet van hoe een IQueryable te implementeren. Bijvoorbeeld (disclaimer: ik heb ze niet in detail bekeken):

- Flickr: http://www.codeproject.com/Articles/20991/LINQ-To-Flickr-Another-IQueryable-Implementation
- Amazon: http://weblogs.asp.net/fmarguerie/archive/2006/06/26/Introducing-Linq-to-Amazon.aspx
- Twitter: http://linqtotwitter.codeplex.com/
- Facebook: http://www.codeproject.com/Articles/34005/LINQ-to-FQL-Facebook-Query-Language

Algemene informatie over custom IQueryable implementaties:

- http://msdn.microsoft.com/en-us/library/bb546158.aspx
- http://msdn.microsoft.com/en-us/library/bb892929%28v=vs.90%29.aspx

'Als het voor Flickr kan, dan moet het ook kunnen voor GAP.' Maar hoe dat juist in zijn werk zal gaan, moet nog verder onderzocht worden.

Als het niet mogelijk is om IQueryable te implementeren, zouden we ook heel de API zelf kunnen implementeren. Bijvoorbeeld via een MVC-toepassing die JSONP oplevert in plaats van html. Maar dat zal ook veel werk vragen, en bovendien verliezen we dan de automagische implementatie van het filteren in de url.

In eerste instantie ('om te spelen') zou het natuurlijk wel mogelijk zijn dat de API gewoon alle leden ophaalt waar de gegeven gebruiker toegang tot heeft (cachen), en daar dan de 'dataservicemagie' op loslaat. Maar dat is uiteraard niet houdbaar als de API ooit meer dan 3 gebruikers heeft :-)

Wat gaat er verder gebeuren?

- Ben zoekt verder op de authenticatie.
- Johan voorziet iets zodat de API op een of andere manier wat data kan opvragen bij de backend, enkel bij wijze van experiment.

Er is een branch gemaakt van de GAP-solution om met een API te experimenteren: https://develop.chiro.be/subversion/cg2/branches/gap-met-api. Verder is er ook de proof of concept https://develop.chiro.be/subversion/cg2/trunk/poc/Chiro.Poc.OData. Ieder die mee wil experimenteren, is van harte uitgenodigd. Aarzel niet om Johan te contacteren als je vragen of problemen hebt.

Varia

Verder hadden we het (off-topic) over:

- De VRT biedt tegenwoordig ook een API aan, maar ze weten nog niet wat ze ermee gaan doen. http://services.vrt.be
- i3wm, een tiling window manager: http://i3wm.org/
- http://buyvm.net, waar je goedkoop VPS'en kunt kopen.
- Ncmpcpp, een ncurses-based mpd-client: http://unkart.ovh.org/ncmpcpp/