WERKGROEP CHIROGROEP		457
15 april 2009	Not.	6

Aanwezig: Tom Haepers, Peter Bertels, Bart Boone, Johan Vervloet
Verontschuldigd: Broes De Cat

ChannelFactory

- Huidige implementatie: zie bijlage 1
- Tommy kan ons code bezorgen voor een statische klasse 'ServiceHelper', die op basis van de Interface, een lambda-expressie en informatie in de web.Config weet hoe de service aangeroepen moet worden, en wat het resultaat is. Je krijgt dan code die er ongeveer zo uitziet: (als ik het me goed herinner)

```
p = ServiceHelper.CallService<IGelieerdePersonenService, GelieerdePersoon>(s =>
s.PersoonOphalen(persoonID))
```

Foutafhandeling

- Huidige implementatie: zie bijlage 2
- De fouten in de businesslayer worden bewaard in 'ViewData.ModelState', maar dat is vermoedelijk de plaats voor binding errors. Voor businesserrors moet het model de interface waarschijnlijk 'IDataErrorInfo' implementeren. Misschien moeten we eerst nog wel eens de documentatie bekijken; ik heb het volgende gevonden: http://www.asp.net/learn/mvc/#MVC_Validation.
- De bedoeling is dat bij een 'mislukte verhuis' een VerhuisFault over de lijn komt, met daarin een lijst van problemen. Op die manier weten we het ook als zowel straatnaam als gemeente fout zijn. Schematisch:

```
class FoutBericht<T> {
       T Foutcode {get; set;}
       string boodschap;}

class BusinessFault<T> {
       List<FoutBericht<T> >  Berichten {get; set;}}

class VerhuisFault: BusinessFault<VerhuisFoutCode> {}
```

Disconnected entity framework

- Bij het verhuizen van personen treedt er in sommige gevallen een exceptie op bij 'AttachObjectGraph'.
- We moeten sowieso bekijken of het niet nuttiger is om in de plaats van gedetachte entity's opnieuw te attachen, de bestaande entities opnieuw op te vragen, en de property's/relaties van die bestaande entity's aan te passen.

Programmeerdag(en) 22 (en 23) mei

- Johan informeert
  - wie er ook zaterdag kan
  - of we kunnen overnachten op Kipdorp
  - wat het budget is voor 'de catering'
- Peter en Johan stellen een lijst van afgelijnde taken op die op de programmeerdag kunnen gebeuren.
- Johan probeert de use case 'verhuizen' zo snel mogelijk afgewerkt te hebben. Als er nog problemen boven komen waarvoor iets beslist moet worden, wordt er nog een vergadering samengeroepen voor de programmeerdag.
- In principe kan op de programmeerdag aan de user interface gewerkt worden, zonder dat alle functionaliteit in de businesslayer geïmplementeerd is. De service kan gewoon zeggen dat de operatie gelukt is, zonder dat er echt iets gebeurt.

Security

- Huidige situatie: zie bijlage 4
- Nakijken of een gebruiker recht heeft om een persoon/… te zien, gebeurt beter in de businesslaag dan in de servicelaag.
- De ServiceSecurityContext is overal beschikbaar, dus ook in het project Cg2.Data. Het is dus niet nodig om de username als aparte parameter door te geven aan de AuthorisatieDao.
- Ivm authorisatie willen we meestal weten of een gebruiker X object Y mag zien. Hiervoor moeten in AuthorisatieDao functies voorzien worden die dat nakijken, en enkel true of false teruggeven (ipv entity's nu). Het is niet de bedoeling dat er iedere keer in de businesslaag moet nagekeken worden of de vervaldatum van een GAV nog niet voorbij is.

Varia

- Het model VerhuisInfo is niet intuitief, in die zin dat de bewoners van het VanAdres in het model aan het NaarAdres gekoppeld zijn. Dat is geen goede zaak; het model moet aangepast worden. (Eventueel een aparte lijst met de bewoners.)
- In de database ontbreken nog indices. In principe moet er al minstens een index staan op elk veld dat een foreign key is.
- De 'Managers' in Cg2.Workers hebben constructors met interfaces voor de data-access, maar ook standaardconstructors. Die standaardconstructors zouden moeten verdwijnen, en in de plaats daarvan moet er een IOC-container komen de 'standaardmanagers' maakt.

Bijlage 1: ChannelFactory (14/4)

Het verschil met vroeger:

Vroeger werd (achter de schermen) svcutil gebruikt, om de servicedefinitie te downloaden, en op basis daarvan 'proxycode' te genereren. Iedere keer dat er iets aan het servicecontract veranderde, moest de code voor de proxy opnieuw gegenereerd worden.

Nu gebeurt er meer automatisch. Een persoon bewaren gebeurt bijv. als volgt:

```
  using (ChannelFactory<IGelieerdePersonenService> cf = new
ChannelFactory<IGelieerdePersonenService>(_endpointConfig))
  {
      IGelieerdePersonenService service = cf.CreateChannel();
      try
      {
          service.PersoonBewaren(p);
      }
      finally
      {
          ((IClientChannel)service).Close();
      }
  }
```

Het contract voor de WCF-service (IGelieerdePersonenService) komt uit het project Cg2.ServiceContracts; dit is de assembly die ook gebruikt wordt door de service-implementatie zelf (in Cg2.Services).

Als er tot voor vandaag iets wijzigde aan Cg2.ServiceContracts, dan bleef de UI-kant zonder problemen compileren, tot dat de proxycode opnieuw gegenereerd werd. Nu is dat niet meer het geval, aangezien IGelieerdePersonenService gedeeld wordt door servicelaag en UI-laag.

Het stuk in 'finally' mag niet achterwege gelaten worden, maar ik krijg die try-finally structuur niet omgezet naar een gewone 'using', omdat IClientChannel.Close enkel aangeroepen kan worden door de IGelieerdePersonenService expliciet naar IClientChannel te casten.

Ik maak ook iedere keer een nieuwe instantie van ChannelFactory<IGelieerdePersonenService> aan. Ik ben er niet zeker van of dit de meest interessante manier van werken is. Misschien is het handiger/sneller om die Channel Factory 1 keer aan te maken, en te blijven gebruiken? Ik ben er niet zeker van, maar ik heb een boek over WCF besteld.

Dit alles maakt de aanroep van een servicemethod nogal uitgebreid. Daarom heb ik dit soort van calls ook gegroepeerd in aparte statische klasses in 'MvcWebApp.ServiceCalls'. Zie bijv:

https://develop.chiro.be/trac/cg2/browser/trunk/Solution/MvcWebApp2/ServiceCalls/GelieerdePersonen.cs

Bijlage 2: Opvangen van 'businesserrors' in de UI (9/4)

De huidige versie van het Chirogroepprogramma (rev 318) reageert op fouten in de businesslaag door het form opnieuw te tonen, met ingevulde waarden, en propere foutboodschappen.

Om het gauw aan het werk te zien, kan je connecteren op de VPN, en surfen naar

http://devserver.chiro.lokaal/MvcWebApp

Je kan de code ook downloaden/updaten uit de svn-repository; tegenwoordig zit het project in https://develop.chiro.be/subversion/cg2/trunk/Solution/

Als je het project wil runnen op je eigen PC (met bijv. een lokaal geinstalleerde Visual Studio), dan zijn twee zaken belangrijk:
 - er moet een vpn-connectie zijn (voor de database op de devserver)
 - Visual Studio moet lopen onder een domain account, anders zal je geen gebruikersrechten krijgen voor de webservice. (Als je met een lokale account aangemeld bent, kan je Visual Studio met je Chirocredentials starten door rechts op het icoontje te klikken, en 'RUN AS...' te kiezen.)

Ik heb de functionaliteit 'verhuizen' geimplementeerd. Dat wil zeggen: een *bestaande* koppeling gelieerde persoon-adres aanpassen. Een nieuwe koppeling maken werkt nog niet.

Als je in de fiche van een persoon op de link 'verhuizen' klikt achter een van de adressen, dan krijg je een lijstje te zien van personen die op het gevraagde adres wonen. Hieruit kies je de mensen die mee verhuizen. Verder kan je een nieuw adres intypen. Nog niks Ajax, gewoon een saai form.

Tik je nu een adres in waarvan de straat- of gemeentenaam niet klopt, dan krijg je een exception in de businesslaag. Die wordt opgevangen door de controller, die ervoor zorgt dat een gepaste foutmelding getoond zal worden. (Via ViewData.ModelState.AddModelError, wat in MVC ingebakken zit.)

Alles wat de gebruiker intikte in het form, krijgt de Controller terug, meteen in een object VerhuisInfo (het model). Je moet het model wel opnieuw aanvullen met de gegevens die uit de database gehaald moeten worden (in dit geval: de andere bewoners van het adres.)

Hoe het precies in elkaar zit, kan je hier zien in de source:
https://develop.chiro.be/trac/cg2/browser/trunk/Solution/MvcWebApp2/Controllers/PersonenController.cs#L131

(Ik ben er niet zeker van of deze manier van werken met Exceptions wel ideaal is, want als nu zowel straat als gemeente ongeldig zijn, krijg je toch maar 1 foutmelding. De exception is al opgeworpen voor de rest gecontrollerd wordt.)

Bijlage 3: Disconnected Entity Framework (9/4)

Het heeft vrij lang geduurd vooraleer ik dit geimplementeerd had. Dat komt niet omdat het zo moeilijk is om het form terug goed te krijgen met de ingevulde gegevens, maar wel omdat ik nog wat heb zitten sukkelen met 'disconnected entity framework'.

Om problemen met disconnected entities te beperken, is het het interessantst dat alle entity's expliciet van de context gedetacht worden door de data access layer. Ik dacht dat dat wel automatisch zou gebeuren als de objectcontext 'gedisposet' wordt, maar dat bleek niet het geval. Een expliciete context.Detach is nodig.

Nu is het blijkbaar zo dat, als je een entity detacht, de navigation properties (meestal? altijd?) verdwijnen. Dat is natuurlijk niet wat we willen, meestal hebben we ook gekoppelde objecten nodig. (Bijv: de persoon gekoppeld aan een gelieerde persoon enz.) (Als je niet expliciet detacht, maar gewoon de context disposet, dan krijg je de gekoppelde objecten *wel* mee. Maar dan krijg je achteraf soms problemen als er zaken geupdatet moeten worden, of bij het opnieuw attachen.)

Je kan dit opvangen door je query's naar het entity framework uit te voeren met 'MergeOption.NoTracking'. Op die manier worden de entity's gewoon niet aan de objectcontext gekoppeld. Zie bijv:

https://develop.chiro.be/trac/cg2/browser/trunk/Solution/Cg2.Data/GelieerdePersonenDao.cs#L119

MergeOption.NoTracking heeft dan weer het nadeel dat dubbele objecten niet herkend worden, en dat op die manier eenzelfde object meerdere keren in je 'resultaatgraaf' voorkomt. Dit heeft als gevolg dat je soms manueel relaties moet verleggen. Dit gebeurt bijvoorbeeld wanneer ik de PersoonsAdressen van eenzelfde adres opvraag:

https://develop.chiro.be/trac/cg2/browser/trunk/Solution/Cg2.Data/AdressenDao.cs#L98

Een ander probleem waarop ik stuitte, was het terug in de database persisteren van een gewijzigde koppeling. Concreet: Voor de verhuis van een persoon, neem ik het PersoonsAdres dat die persoon met zijn adres koppelt, en wil ik dat PersoonsAdres.Adres wijst naar het nieuwe adres. Dit kreeg ik niet gemakkelijk in orde. Na wat zoeken op internet vond ik dit artikel:

http://www.codeproject.com/KB/architecture/attachobjectgraph.aspx

Dit artikel bevat code die een 'entity graph' opnieuw aan een context attacht. Je definieert welke entity's geattacht moeten worden, en van welke entity's de wijzigingen meegenomen moeten worden, van dewelke niet. Ik heb die code toegevoegd aan onze solution, en ik gebruik ze bij het bewaren van een adres met z'n koppelingen.

https://develop.chiro.be/trac/cg2/browser/trunk/Solution/Cg2.Data/AdressenDao.cs#L30

Er valt hier nog wel wat over te zeggen. Om de relaties tussen entity's te wijzigen, gaat AttachObjectGraph de huidige status van de database bekijken. Nog niet-bestaande relaties worden bijgemaakt, en (!) verdwenen relaties worden verwijderd.

Dat laatste is waarschijnlijk niet helemaal wat we willen: het geeft concurrency problemen als twee personen tegelijk een verschillend adres toevoegen. Het toegevoegde adres van de eerste zal dan opnieuw verdwijnen. (Voor wijzigingen is er geen probleem, omdat we met timestamps werken.)

Waarschijnlijk zullen we AttachObjectGraph wat moeten aanpassen, zodat die naar de timestamp kijkt, en naar de property 'TeVerwijderen' (die nog aan IBasisEntiteit toegevoegd moet worden.)

In het beste geval biedt een volgende versie van Entity Framework een elegantere oplossing voor heel deze problematiek.

Nog een laatste ding dat het opmerken waard is: om de 'CheckBoxList' te genereren met de bewoners die al dan niet mee kunnen verhuizen, heb ik een HtmlHelper gedownload van http://blogs.msdn.com/miah/archive/2008/11/10/checkboxlist-helper-for-mvc.aspx

Bijlage 4: Security (18/3)

De webapplicatie authenticeert de gebruikers tegen AD. De applicatie neemt vervolgens de identiteit van de geauthenticeerde gebruiker over. Als de webapplicatie dan de WCF-service aanroept, gebruikt deze de credentials van de geauthenticeerde gebruiker. Zo weet de servicelaag met wie ze te doen heeft. (De WCF-service is zodanig geconfigureerd dat enkel users uit de groep g-developers-chirogroep ze kunnen aanroepen. Deze groep heb ik voorlopig hardgecodeerd in een settingsklasse; zie https://develop.chiro.be/trac/cg2/browser/trunk/poc/ef2/WebServices/Settings.cs)

Vooraleer de servicelaag nu de relevante worker aanroept, vraagt de servicelaag eerst aan de businesslaag of de gebruiker het wel recht heeft om de gevraagde operatie uit te voeren. Is dat niet het geval, wordt een exception 'gethrowd'.

Dit is geimplementeerd in de functie PaginaOphalenMetLidInfo van de GelieerdePersonenService, zie https://develop.chiro.be/trac/cg2/browser/trunk/poc/ef2/WebServices/GelieerdePersonenService.svc.cs#L37

(De service laag neemt de identiteit van de caller niet over, en de data access layer gebruikt een vaste connectie, die voldoende rechten heeft voor alle mogelijke operaties. Op die manier wordt de connection pooling van .NET op een goeie manier gebruikt.)

Op het niveau van de webapplicatie zelf, heb ik nog niets i.v.m. authorisatie geimplementeerd. Zolang de webapplicatie geen 'adminfunctionaliteit' heeft, en er geen read-only users zijn, is dat ook niet nodig: alle GAV's moeten heel de app kunnen gebruiken.

Deze versie van de applicatie is gedeployd op http://devserver.chiro.lokaal/MvcWebApp. Als je geen pass-throughauthenticatie kan gebruiken, zal er een gebruikersnaam en wachtwoord gevraagd worden. Dat zijn je credentials voor het Chironetwerk, maar let er wel op dat je je gebruikersnaam prefixt met 'KIPDORP\'.

Ik heb jullie allemaal GAV gemaakt van de groep 310, die door het testproject wordt gebruikt (zie de tabellen auth.Gav en auth.Gebruikersrecht in de database.) Als je in auth.Gebruikersrecht je koppeling met groep 310 verwijdert, dan zal je zien dat je een lelijke foutmelding krijgt als je probeert http://devserver.chiro.lokaal/MvcWebApp/Personen/Index te 'accessen'. (Fatsoenlijke exception handling is nog niet geimplementeerd.) Voorlopig is enkel 'PaginaOphalenMetLidInfo' beveiligd, bijgevolg zijn de andere pagina's van de app nog wel toegankelijk (als je geauthenticeerd bent uiteraard.)