Chatty

Conception

Nicolas Sabella / Julien Cazeneuve / Chahine Mouhamad

MOTS-CLES-SOUS-CE-FORMAT

12 Pages

01/06/2014

Multimédia
Technologies de l'Information

EPITA
ÉCOLE D'INGÉNIEURS EN INFORMATIQUE

Propriétés du document

Auteur	Nicolas Sabella / Julien Cazeneuve / Chahine Mouhamad
Version	1.0
Nombre de pages	11
Références	MOTS-CLES-SOUS-CE-FORMAT
Nom fichier	Document3

Historique du document

Date de révision	Version	Auteur	Changements
	0.0	TRIGRAM	

Table des matières

COUCHE SERVICE

WCF

Windows Communication Foundation (WCF) est le modèle de programmation unifié de Microsoft permettant de générer des applications orientées service. Il permet aux développeurs de générer des solutions transactionnelles sécurisées et fiables qui s'intègrent à plusieurs plateformes et interagissent avec les investissements existants.

WCF comprend l'ensemble de fonctionnalités suivantes :

- Orientation services
- Interopérabilité
- Modèles de messages variés
- Métadonnées de service
- Contrats de données
- Sécurité
- Transports et encodages variés
- Messages fiables et mise en file d'attente
- Transactions
- Prise en charge d'AJAX et de REST
- Extensibilité

NETTCPBINDING

netTcpBinding est utilisé pour communiquer avec les clients WCF dans un environnement intranet et assure la sécurité des transports et de l'authentification Windows par défaut. netTcpBinding offre des performances améliorées sur une liaison HTTP. Le service WCF avec netTcpBinding peut être consommé par une application via l'utilisation d'une référence de service.

DUPLEX BINDING

Un contrat de service duplex est un modèle d'échange de messages dans lequel les deux points de terminaison peuvent envoyer indépendamment des messages à l'autre. Un service duplex peut, par conséquent, renvoyer des messages au point de terminaison client, en fournissant un comportement de type événement. La communication duplex se produit lorsqu'un client se connecte à un service et lui fournit un canal sur lequel il peut lui renvoyer des messages.

PATRON DE CONCEPTION : OBSERVATEUR

Pour réaliser l'interaction entre le client et le serveur nous implémenterons le patron de conception observateur afin de répondre à nos besoins.

COUCHE DONNEE

EF 6

Pour réaliser la liaison avec notre base de données nous allons utiliser Entity Framework 6, ORM (object-relational mapping), il s'agit technique de programmation informatique qui crée l'illusion d'une base de données orientée objet à partir d'une base de données relationnelle en définissant des correspondances. Entity Framework 6 permet d'éviter d'écrire des requête SQL au sein du code mais aussi de faire abstraction de la base de donnéees.

EMITMAPPER

Afin de mapper rapidement et facilement les objets envoyés par la couche donnée vers des objets de la couche Dbo on utilisera EmitMapper. On a choisi EmitMapper du fait que ce dernier

- est beaucoup plus performant qu'Automapper
- minize les opérations de boxing-unboxing et les appels supplémentaire lors du mapping
- est facile à utiliser
- est flexible

SQL SERVER

Sql Server est un système de gestion de base de données (SGBD) développé et commercialisé par Microsoft. Du fait de son aspect multibase, SQL Server dispose d'une sécurité à deux niveaux : niveau serveur, par le biais des comptes de connexion et niveau base, par le biais des utilisateurs SQL.

TABLES

User

Id	int
Username	varchar(255)
Firstname	varchar(255)
LastName	varchar(255)
Email	varchar(255)
Password	varchar(255)
Thumbnail(photo de profil)	text
IsEnable(banni ou non)	boolean
DepartmentId	int

Message

Id	bigint
UserId(sender)	int
Content	text
Time	datetime

Department

Id	int
Name	varchar(255)

Group

Id	int
Name	varchar(255)

UserId(Admin du groupe)	int

GroupMessages

(relation many-to-one : un group a plusieurs messages)

GroupId	int
MessageId	Bigint

GroupUsers

(relation many-to-one : un group a plusieurs user)

GroupId	int
UserId	int

UserContacts

(relation many-to-one : un user a plusieurs contact : user)

UserId	int
ContactId	int

Invitation

Id	int
FromUserId	int
ToUserId	int
Content	text

APPLICATION CLIENT

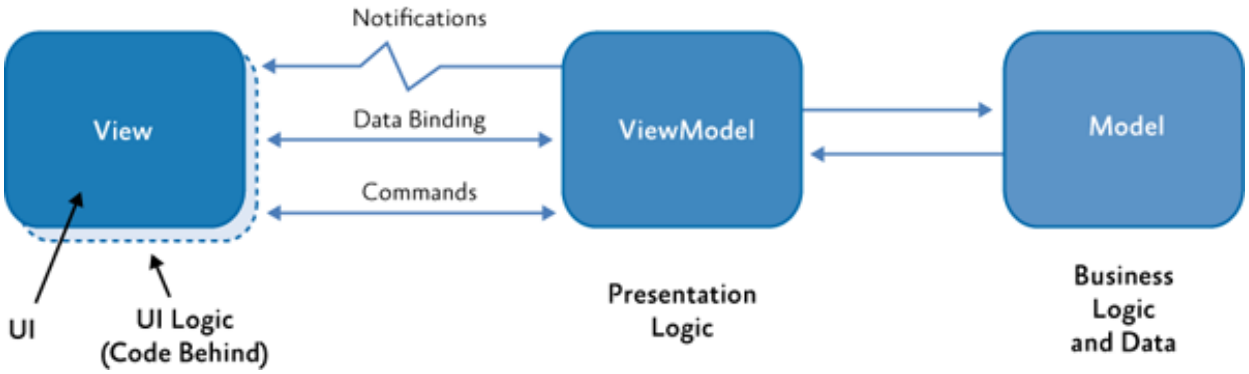
WPF

L'application client sera réalisée en WPF.

PATTERN MVVM

Ce pattern a spécialement été conçu pour améliorer la séparation entre les données et la vue qui les affichent. Le lien entre la vue et le modèle de données est fait par des mécanismes de binding. Pour rappel, le binding est un mécanisme qui permet de faire des liaisons entre des données de manière dynamiques. Ce qui veut dire que si A et B sont lié, le fait de modifier A va être répercuté sur B et inversement.

D'une manière générale, le modèle modèle-vue-ViewModel tente de gagner à la fois les avantages de la séparation du développement fonctionnel fournies par MVC ainsi que tirer parti des avantages du databinding. Alors que le modèle a été mis au point à Microsoft, en tant que concept pur, il est indépendant de toute application donnée. Comme tel, le modèle peut être utilisé dans n'importe quelle langage de programmation et avec n'importe quel Framework qui fournit une fonctionnalité de databinding déclaratives.



Voyons de plus prêt ce que contient MVVM :

- Model (Modèle en français) : le modèle contient les données. Les données proviennent de la base de données via la couche service.
- View (Vue en français) : la vue correspond à ce qui est affiché. La vue contient les différents composants graphiques (boutons, liens, listes) ainsi que le texte.
- ViewModel (Vue-Modèle en français) : ce composant fait le lien entre le modèle et la vue. Il s'occupe de gérer les liaisons de données et les éventuelles conversions. C'est ici qu'intervient le binding.

BACKEND

Le backend, soit l'interface d'administration de la solution Chatty, sera d'une part développé à l'aide du framework web ASP.NET et selon le modèle MVC qu'offre le framework.

La version utilisée sera ASP.NET MVC 5.

Le modèle MVC possède ces composantes, clairement similaires au MVVM :

- Modèle : Encapsule le cœur fonctionnel de l'application, le domaine logique.
- Vue : les données sont envoyées, par le modèle, à la vue qui les présente à l'utilisateur.
- Contrôleur : reçoit les données et les transmets au modèle ou à la vue.

Les différences avec le MVVM est que le modèle est pleinement exposé à la vue grâce au contrôleur, qui agit comme coordinateur entre donc la vue et le modèle. Tandis qu'en MVVM, la vue n'a aucune information sur le modèle et n'affiche que les données nécessaires selon les bindings faits avec le ViewModel, ce qui permet un couplage faible entre nos objets.

Le backend va être de type SPA (Single Page Application), il y a donc nécessité de pouvoir effectuer des actions sur notre page de manière asynchrone.

On va utiliser dans cette optique le framework Javascript appelé AngularJS.

Ce framework impose également une architecture MVC côté client, un de ses points forts est le databinding bi-directionnel, c'est-à-dire qu'on va pouvoir synchroniser automatiquement nos vues côté client et les modèles associés.

Les données à aller chercher depuis la SPA seront fetchables depuis des webservices RESTful à l'aide de WebAPI, composante d'ASP.NET MVC. Elles seront traitées en JSON.

Exemple : On effectue une recherche d'utilisateurs via un formulaire, qui effectue une requête asynchrone renvoyant des résultats via un webservice de la WebAPI, on met à jour la liste de résultats côté client, et ils s'affichent automatiquement dans la vue.

La SPA sera réalisée en HTML5/CSS.

SCHEMA DE CONCEPTION