Developpez

Le Mag

Édition de février - mars 2013.
Numéro 44.
Magazine en ligne gratuit.
Diffusion de copies conformes à l'original autorisée.
Réalisation : Alexandre Pottiez
Rédaction : la rédaction de Developpez
Contact : magazine@redaction-developpez.com

Sommaire

Éditorial

La rédaction se plie une fois encore en quatre pour vous permettre de lire le meilleur des publications dans toutes vos technologies préférées.

Profitez-en bien !

La rédaction

Article Excel

Les filtres avancés ou élaborés dans Excel, les paramétrer et les utiliser

Outil puissant et finalement très peu connu par les utilisateurs le filtre élaboré permet de filtrer des données avec plus de possibilités que le filtre simple dont on atteint très vite ses limites.

par **Philippe Tulliez**
Page 2

Article Delphi

Envoyer des chaînes ou des structures par PostMessage

Ce tutoriel a pour but d'expliquer une façon simple d'envoyer des chaînes de caractères et des structures complexes en asynchrone par PostMessage.

par **Andnotor**
Page 51

Les filtres avancés ou élaborés dans Excel - Paramétrer et utiliser les filtres avancés

Outil puissant et finalement très peu connu par les utilisateurs le filtre élaboré permet de filtrer des données avec plus de possibilités que le filtre simple dont on atteint très vite ses limites.

En plus de filtrer les données sur place, il permet l'exportation de celles-ci vers une autre feuille ou un autre classeur. Son exploitation en VBA offre de belles perspectives de développement.

J'espère que la lecture de ce tutoriel vous permettra de le découvrir ou d'en apprendre plus sur ses possibilités.

1. Introduction

Vous voulez extraire des enregistrements d'une table de données selon certains critères : s'ils sont relativement simples, le filtre automatique répondra parfaitement à votre attente, si par contre vos critères sont plus complexes (champs calculés, suite de critères logiques **ET** et **OU**\x{0085}) vous atteindrez vite les limites du filtre automatique.

1.1. Glossaire

Les filtres avancés sont connus aussi sous le nom de filtres élaborés. Nous emploierons donc indifféremment ces deux termes.
Table de données : la première ligne d'une table de données Excel doit contenir les noms des champs, appelés aussi étiquettes ; les lignes, de la deuxième à la dernière, contiennent toutes les données (par exemple les renseignements concernant les clients) : ces lignes sont appelées "enregistrements".
Chaque colonne contient, pour chaque enregistrement, les données correspondant à l'étiquette de la colonne.
Enregistrement : ligne contenant tous les éléments spécifiques d'un objet déterminé et unique (par exemple les données d'un client, d'un article\x{0085}).
Étiquette : chaque cellule de la première ligne de la table de données, nommant le contenu de la colonne (par exemple : "Nom", "Prénom", Numéro", etc.).

2. Étape préliminaire à la préparation d'un filtre avancé

Pour utiliser un filtre élaboré, il faut au moins une table de données et une zone de critères.

1. La table de données doit avoir en première ligne les étiquettes de colonne.
2. La zone de critères doit avoir en première ligne des étiquettes de colonnes et au moins une ligne avec un ou plusieurs critères sauf pour filtrer les doublons (ce sujet sera abordé dans un chapitre ultérieur).

Partons de cette table de données qui va nous permettre de tester les filtres élaborés sans trop de difficultés.
Déplaçons ce tableau jusqu'à la ligne 5 et copions les étiquettes de ligne de ce tableau sur la 1re ligne de la feuille Excel.

	A	B	C	D	E	F	G
1	Prénom	Lieu	Naissance	Math	Français	Sciences	Moyenne
2							
3							
4							
5	Prénom	Lieu	Naissance	Math	Français	Sciences	Moyenne
6	Tanguy	Bruxelles	5/02/1998	8	6	8	7,3
7	Magali	Mons	11/11/1997	9	7	8	8,0
8	Isabelle	Liège	12/04/1997	6	7	5	6,0
9	Stéphane	Bruxelles	13/07/1997	7	9	8	8,0
10	Alain	Liège	3/12/1997	10	8	9	9,0
11	Philippe	Bruxelles	2/03/1998	8	8	7	7,7
12	Marie	Namur	30/06/1997	7	10	8	8,3

les étiquettes de la zone de critères doivent avoir la même orthographe que les étiquettes de la table de données.

3. Comment activer le filtre élaboré

- Excel 2003 Menu [Données] - Filtrer - Filtre élaboré.
- Excel 2007 et suivants : Onglet [**Données**], groupe Trier et filtrer, commande **Avancé.**

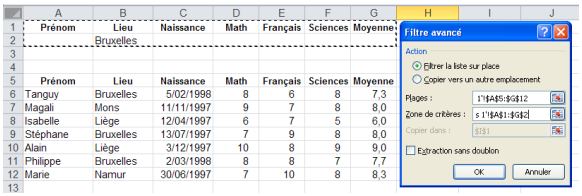
Ensuite la boîte de dialogue apparaît, la zone Plages remplie si le curseur se trouve dans une des cellules de la table des données.

4. Quelques exemples simples

4.1. Premier exemple

Commençons par un cas simple : filtrer les lignes (les enregistrements) où la colonne **Lieu** est égale à "Bruxelles".
Dans la cellule **B2** qui se trouve juste en dessous de l'étiquette **Lieu**, nous introduisons la valeur "Bruxelles".

Dans le groupe Trier et filtrer de l'onglet **[Données]**, cliquons sur la commande **Avancé.** La boîte de dialogue **Filtre avancé** apparaît.



Analysons chaque option de cette boîte.

- **Filtrer la liste sur place** : filtre directement dans la zone de cellules d'Excel où sont placées les données de départ et qui sont déclarées par l'option **Plages** (dans notre exemple **A5:G12**).
- **Copier vers un autre emplacement** : permet de créer la liste filtrée vers un autre emplacement défini par l'option **Copier dans**.
- **Plages** : la zone à filtrer.
- **Zone de critères** : désigne la plage de cellules où nous avons inséré nos critères de filtrage (dans notre exemple **A2:G2**).
- **Copier dans** : désigne la cellule ou la plage de cellules à partir de laquelle la copie des lignes filtrées se fera. Cette option n'est accessible que si l'option **Copier vers un autre emplacement** est cochée.
- **Extraction sans doublon** : permet de ne pas afficher les données en double soit avec le filtre sur place, soit en copiant vers un autre emplacement.

Nous optons donc pour le filtre sur place en prenant comme plage **A5:G12** et comme zone de critères **A1:G2**. Confirmons par **OK** et voyons le résultat.

	A	B	C	D	E	F	G
1	Prénom	Lieu	Naissance	Math	Français	Sciences	Moyenne
2		Bruxelles					
3							
4							
5	Prénom	Lieu	Naissance	Math	Français	Sciences	Moyenne
6	Tanguy	Bruxelles	5/02/1998	8	6	8	7,3
9	Stéphane	Bruxelles	13/07/1997	7	9	8	8,0
11	Philippe	Bruxelles	2/03/1998	8	8	7	7,7
13							

Nous pouvons constater que les numéros des lignes filtrées sont de couleur bleue et que les lignes 7, 8, 10 et 12 sont masquées : en effet, la valeur contenue dans ces lignes est différente du critère choisi ("Bruxelles").
Ce résultat aurait bien entendu pu être obtenu par un filtre simple.
Pour afficher à nouveau les données, il suffit de cliquer sur la commande **Effacer** qui se trouve dans le même groupe que la commande **Filtre élaboré**.

4.2. Deuxième exemple

Filtrons maintenant les enregistrements pour lesquels le lieu est "Bruxelles" ET dont la date de naissance est supérieure ou égale au 1er janvier 2008.
Ajoutons donc en cellule **C2** la valeur >=01/01/2008. Exécutons ensuite la même commande en choisissant les mêmes options.

	A	B	C	D	E	F	G
1	Prénom	Lieu	Naissance	Math	Français	Sciences	Moyenne
2		Bruxelles	>=01/01/1998				
3							
4							
5	Prénom	Lieu	Naissance	Math	Français	Sciences	Moyenne
6	Tanguy	Bruxelles	5/02/1998	8	6	8	7,3
11	Philippe	Bruxelles	2/03/1998	8	8	7	7,7
13							

Nous constatons qu'il n'y a plus que deux enregistrements qui répondent à ces deux critères combinés.

4.3. Troisième exemple

Appliquons ensuite un filtre en ajoutant un critère : (Lieu=Bruxelles et Date de naissance>=01/01/1998) **OU** (Moyenne>7,5).
Pour obtenir le résultat escompté, ajoutons en **G3** (colonne **Moyenne)** la valeur **>7,5**.
Dans la boîte de dialogue, modifions la zone de critères en mettant **A1:G3**.

	A	B	C	D	E	F	G
1	Prénom	Lieu	Naissance	Math	Français	Sciences	Moyenne
2		Bruxelles	>=01/01/1998				
3							>7,5
4							
5	Prénom	Lieu	Naissance	Math	Français	Sciences	Moyenne
6	Tanguy	Bruxelles	5/02/1998	8	6	8	7,3
7	Magali	Mons	11/11/1997	9	7	8	8,0
9	Stéphane	Bruxelles	13/07/1997	7	9	8	8,0
10	Alain	Liège	3/12/1997	10	8	9	9,0
11	Philippe	Bruxelles	2/03/1998	8	8	7	7,7
12	Marie	Namur	30/06/1997	7	10	8	8,3

En conclusion, nous constatons que placer des critères sur une même ligne équivaut à la fonction **ET** ; par contre, si nous utilisons plusieurs lignes, cela équivaut à la fonction **OU**

4.4. Quatrième exemple

Cet exemple est pratiquement le même que le précédent sauf qu'ici dans la deuxième ligne (**OU**), nous ajouterons Bruxelles dans le critère **Lieu**.
Ce qui signifie que nous filtrons les enregistrements avec comme critères :
(Lieu =**Bruxelles**) **ET** (Naissance>=01/01/1998 **OU** Moyenne>7,5),
ce qui peut s'écrire également
(Lieu=**Bruxelles ET** Date de naissance>=01/01/1998) **OU** (Lieu=**Bruxelles ET** Moyenne>**7,5**).

	A	B	C	D	E	F	G
1	Prénom	Lieu	Naissance	Math	Français	Sciences	Moyenne
2		Bruxelles	>=01/01/1998				
3		Bruxelles					>7,5
4							
5	Prénom	Lieu	Naissance	Math	Français	Sciences	Moyenne
6	Tanguy	Bruxelles	5/02/1998	8	6	8	7,3
9	Stéphane	Bruxelles	13/07/1997	7	9	8	8,0
11	Philippe	Bruxelles	2/03/1998	8	8	7	7,7

5. Le filtre élaboré avec plusieurs critères

Dans les exemples précédents, nous avons utilisé un seul critère de filtre par colonne; dans les suivants, nous utiliserons simultanément plusieurs critères.
Par exemple, nous cherchons à filtrer les données de personnes dont les moyennes de cotes se situent entre 7,5 et 9.

Précision importante : jusqu'à présent, nous avions dans

la zone des critères, le même nombre de colonnes que la table de données. Cela n'est absolument pas indispensable, seules les colonnes et leurs étiquettes concernées par les critères doivent être présentes.

Nous allons donc utiliser deux colonnes de critères portant la même étiquette **Moyenne**, comme le montre l'exemple ci-dessous.
Dans la boîte de dialogue, l'option Zone de critères fera donc référence aux cellules **A1:B2**.

	A	B	C	D	E	F	G
1	**Moyenne**	**Moyenne**					
2	>=7,5	<=9					
3							
4							
5	**Prénom**	**Lieu**	**Naissance**	**Math**	**Français**	**Sciences**	**Moyenne**
7	Magali	Mons	11/11/1997	9	7	8	8,0
9	Stéphane	Bruxelles	13/07/1997	7	9	8	8,0
10	Alain	Liège	3/12/1997	10	8	9	9,0
11	Philippe	Bruxelles	2/03/1998	8	8	7	7,7
12	Marie	Namur	30/06/1997	7	10	8	8,3

Si nous souhaitons appliquer un critère supplémentaire, par exemple, les personnes de Bruxelles ayant une moyenne entre 7 et 9, nous ajouterons une colonne de critères et dans la boîte de dialogue, l'option Zone de critères fera donc référence aux cellules **A1:C2**, comme l'illustre l'exemple ci-dessous.

	A	B	C	D	E	F	G
1	**Moyenne**	**Moyenne**	**Lieu**				
2	>=7,5	<=9	Bruxelles				
3							
4							
5	**Prénom**	**Lieu**	**Naissance**	**Math**	**Français**	**Sciences**	**Moyenne**
9	Stéphane	Bruxelles	13/07/1997	7	9	8	8,0
11	Philippe	Bruxelles	2/03/1998	8	8	7	7,7

6. Les critères calculés

6.1. Exemples de critère calculé

Un des gros avantages du filtre élaboré, fort méconnu, est l'utilisation de critères calculés.
Pour les utiliser, entrons une formule en lieu et place d'une constante et plaçons comme étiquette de colonne un nom que l'on ne retrouve pas comme en-tête dans la table de données.
Ainsi, si l'on souhaite filtrer les données des personnes nées en 1997, nous placerons une colonne avec comme étiquette **AnneeNaiss** et à la ligne suivante la formule **=ANNEE(C6)=1997**.

A2 : =ANNEE(C6)=1997

	A	B	C	D	E	F	G
1	**AnneeNaiss**						
2	FAUX						
3							
4							
5	**Prénom**	**Lieu**	**Naissance**	**Math**	**Français**	**Sciences**	**Moyenne**
6	Tanguy	Bruxelles	5/02/1998	8	6	8	7,3
7	Magali	Mons	11/11/1997	9	7	8	8,0
8	Isabelle	Liège	12/04/1997	6	7	5	6,0
9	Stéphane	Bruxelles	13/07/1997	7	9	8	8,0

Nous constatons que la cellule **A2** qui contient la formule **=ANNEE(C6)=1997** et qui fait référence à la cellule **C6** renvoie **FAUX :** en effet la valeur de la cellule en **C6** est le 5 février 1998 donc année 1998.

A2 : =ANNEE(C6)=1997

	A	B	C	D	E	F	G
1	**Année**						
2	FAUX						
3							
4							
5	**Prénom**	**Lieu**	**Naissance**	**Math**	**Français**	**Sciences**	**Moyenne**
7	Magali	Mons	11/11/1997	9	7	8	8,0
8	Isabelle	Liège	12/04/1997	6	7	5	6,0
9	Stéphane	Bruxelles	13/07/1997	7	9	8	8,0
10	Alain	Liège	3/12/1997	10	8	9	9,0
12	Marie	Namur	30/06/1997	7	10	8	8,3

En conclusion : lorsque nous utilisons des critères calculés dans les filtres élaborés :
l'étiquette de colonne doit porter un nom différent de celui d'une étiquette de la table de données ;
le critère doit être une formule ou une suite de formules imbriqués qui doit renvoyer **VRAI** ou **FAUX ;**
le test logique DOIT être effectué sur une ou plusieurs cellules de la première ligne de la table de données.

Nous pourrions bien entendu ajouter des critères calculés.
Par exemple, filtrons les enregistrements des personnes nées en 1997 et dont la cote en math est égale ou supérieure à la moyenne des points obtenus en math.
Ajoutons une colonne critère dont l'étiquette sera **MoyMath** et la formule **=D6>MOYENNE($D$6:$D$12)**. Sachant que la moyenne obtenue en math pour l'ensemble des personnes enregistrées est de **7,8**, le résultat final est de deux enregistrements filtrés.

	A	B	C	D	E	F	G
1	**AnneeNaiss**	**MoyMath**					
2	FAUX	VRAI					
3							
4							
5	**Prénom**	**Lieu**	**Naissance**	**Math**	**Français**	**Sciences**	**Moyenne**
7	Magali	Mons	11/11/1997	9	7	8	8,0
10	Alain	Liège	3/12/1997	10	8	9	9,0

On aurait pu bien sûr dans ce cas se limiter à une seule colonne contenant le critère calculé avec, par exemple, comme étiquette **AnSupMoy** et comme formule :
=ET(ANNEE(C6)=1997;D6>=MOYENNE($D$6:$D$12))
ou encore
=(ANNEE(C6)=1997)*(D6>=MOYENNE($D$6:$D$12))
.

6.2. Pour bien comprendre le principe du critère calculé

Petit rappel : en **A2** la formule est **=ANNEE(C6)=1997**, en **B2** nous avons **=D6>MOYENNE($D$6:$D$12).**
Saisissons ces mêmes formules en **I6** & **J6** et recopions-les vers le bas, soit la plage **I6:J12**.

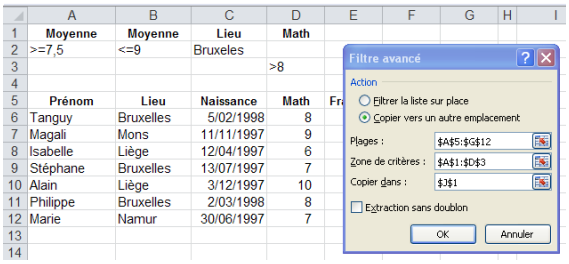
	A	B	C	D	E	F	G	H	I	J
1	**AnneeNaiss**	**MoyMath**								
2	FAUX	VRAI								
3										
4										
5	**Prénom**	**Lieu**	**Naissance**	**Math**	**Français**	**Sciences**	**Moyenne**		**AnneeNaiss**	**MoyMath**
6	Tanguy	Bruxelles	5/02/1998	8	6	8	7,3		FAUX	VRAI
7	Magali	Mons	11/11/1997	9	7	8	8,0		**VRAI**	**VRAI**
8	Isabelle	Liège	12/04/1997	6	7	5	6,0		VRAI	FAUX
9	Stéphane	Bruxelles	13/07/1997	7	9	8	8,0		VRAI	FAUX
10	Alain	Liège	3/12/1997	10	8	9	9,0		**VRAI**	**VRAI**
11	Philippe	Bruxelles	2/03/1998	8	8	7	7,7		FAUX	VRAI
12	Marie	Namur	30/06/1997	7	10	8	8,3		VRAI	FAUX

Nous constatons que seules deux lignes renvoient **VRAI** dans les deux colonnes **I & J**.

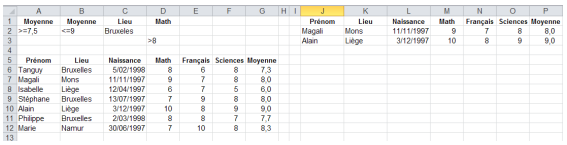
7. Exporter les données filtrées

Un grand avantage du filtre élaboré est sa capacité à

exporter des données filtrées vers un autre emplacement.
L'exportation des données filtrées se prépare de la même manière que ce qui a été décrit plus haut.
Il suffit de cocher l'option Copier vers un autre emplacement et ensuite référencer dans la zone Copier dans, la cellule à partir de laquelle doivent s'exporter les enregistrements filtrés, comme le montre l'illustration.



Les éléments ainsi filtrés se retrouvent bien à partir de **$J$1**.



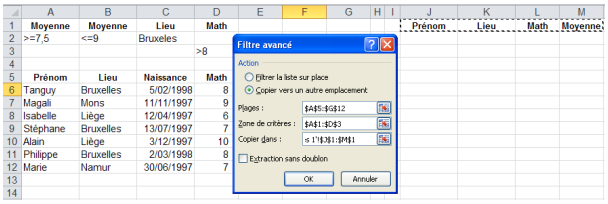
Il y a deux possibilités d'exporter les données filtrées : soit l'entièreté des champs de la table de données, soit certains champs seulement.

7.1. Exportation de toutes les données filtrées

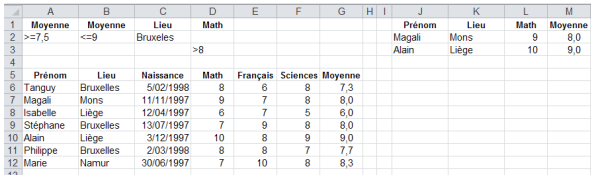
C'est l'option par défaut que l'on retrouve dans l'exemple ci-dessus : nous avons simplement indiqué la référence de départ (**$J$1**) dans le champ Copier dans : de la boîte de dialogue **Filtre avancé**.

7.2. Exportation partielle des données filtrées

L'exportation partielle demande une petite préparation.
Prenons comme hypothèse que nous ne souhaitons avoir que les colonnes Prénom, Lieu, Math et Moyenne des enregistrements ainsi filtrés.
Copions les étiquettes de ces colonnes de **J1** à **M1** et ensuite dans la zone Copier dans de la boîte de dialogue **Filtre avancé**, sélectionnons les cellules **J1:M1**.



Après confirmation, les données filtrées sont exportées.



L'autre avantage de l'exportation partielle est la présentation dans un ordre différent des colonnes de celui de la table de données initiale. Il suffit de placer les étiquettes dans l'ordre souhaité.

7.3. Exportation vers une autre feuille que la table de données

Curieusement, il est impossible de sélectionner une autre feuille dans la zone Copier dans de la boîte de dialogue **Filtre avancé**. Même l'utilisation d'une plage nommée ne fonctionne pas.
Pour pallier ce problème, il faut lancer le filtre élaboré depuis la feuille d'où l'on veut exporter (il faudrait plutôt dire importer), la liste filtrée d'une table se trouvant sur une autre feuille.

8. Exportation sans doublon

8.1. Principe

Cette fonctionnalité est surtout intéressante lorsqu'il s'agit de ne prendre que les éléments uniques d'une colonne, surtout depuis la version 2007 d'Excel qui intègre un outil qui permet de supprimer les doublons.
Par exemple, nous aimerions avoir la liste des villes utilisées dans la colonne B (étiquette **Lieu**).
Ouvrir la boîte de dialogue **Filtre avancé** avec les options Copier vers un autre emplacement et Extraction sans doublon cochées, Plages : **$B$1:$B$8**, rien dans la zone de critères, Copier dans : **$I$1**.

Prénom	Lieu	Naissance	Math	Français	Sciences	Moyenne		Lieu
Tanguy	Bruxelles	5/02/1998	8	6	8	7,3		Bruxelles
Magali	Mons	11/11/1997	9	7	8	8,0		Mons
Isabelle	Liège	12/04/1997	6	7	5	6,0		Liège
Stéphane	Bruxelles	13/07/1997	7	9	8	8,0		Namur
Alain	Liège	3/12/1997	10	8	9	9,0		
Philippe	Bruxelles	2/03/1998	8	8	7	7,7		
Marie	Namur	30/06/1997	7	10	8	8,3		

Mais bien entendu si l'on a des lignes en double, on peut parfaitement faire une exportation sans doublon.

8.2. Ce qu'il faut savoir dans l'exportation des doublons

Important : en ce qui concerne l'exportation, le filtre sur les doublons dépend des étiquettes de colonnes de la zone d'exportation.
Dans la table de données ci-dessous, nous avons trois enregistrements avec des doublons sur l'ensemble des champs à l'exception du champ voiture.
Observons le résultat de l'exportation sans doublon des femmes habitant un Studio dans les deux exemples ci-dessous.
Le premier est le résultat de l'exportation sur trois champs (**H4:J4**) , l'autre sur quatre (**H11:K11**).

Logement	Sexe
Studio	F

Prénom	Sexe	Enfant	Situation	Logement	Voiture
Tanguy	M		Célibataire	Studio	Citroën
Magali	F	2	Couple	Loft	Toyota
Isabelle	F		Couple	Studio	Volvo
Stéphane	M	3	Célibataire	Loft	
Isabelle	F		Couple	Studio	Volvo
Philippe	M		Couple	Studio	
Marie	F	1	Célibataire	Loft	Renault
Eric	M		Couple	Studio	Volvo
Julien	M	2	Célibataire	Maison	Mercedes
Isabelle	F		Couple	Studio	Mercédes
Aloys	M	1	Couple	Maison	Volkswagen
Marie-Tina	F	2	Célibataire	Appartement	Mazda
Vanessa	F	3	Couple	Appartement	
Benedicte	F	2	Couple	Appartement	Peugeot
Sandra	F	1	Célibataire	Studio	

Prénom	Sexe	Logement
Isabelle	F	Studio
Sandra	F	Studio

Prénom	Sexe	Logement	Voiture
Isabelle	F	Studio	Volvo
Isabelle	F	Studio	Mercédes
Sandra	F	Studio	

9. Quelques exemples

Utilisons d'autres données pour les exemples qui vont suivre.

	Prénom	Date	Enfant	Logement	Voiture
2	Tanguy	8/01/2012		Studio	Citroën
3	Magali	1/07/2011	2	Loft	Toyota
4	Isabelle	10/05/2011		Studio	Volvo
5	Stéphane	12/12/2011	3	Loft	
6	Alain	15/04/2012	1	Studio	Ford
7	Philippe	23/06/2011		Studio	
8	Marie	18/08/2011	1	Loft	Renault
9	Eric	14/05/2011		Studio	Volvo
10	Julien	11/07/2011	2	Maison	Mercedes
11	Sandra	25/10/2011	3	Maison	LandRover
12	Aloys	4/03/2012	1	Maison	Volkswagen
13	Marie-Tina	3/11/2012	2	Studio	Mazda
14	Vanessa	19/03/2012	3	Appartement	
15	Benedicte	15/04/2011	2	Appartement	Peugeot
16	Sandra	28/12/2012	1	Studio	

9.1. L'utilisation des caractères génériques

Comme dans les recherches d'Excel, il est possible d'utiliser les caractères génériques, astérisque (*) et point d'interrogation (?).

Petit rappel : le point d'interrogation (?) remplace un caractère quelle que soit sa valeur, l'astérisque (*) remplace 0 ou plusieurs caractères quelles que soient leurs valeurs. Si la valeur à filtrer contient un des caractères **?** ou *, il faut les faire précéder du caractère **~.**

Ainsi, filtrons les personnes dont le prénom contient la lettre **L**.

	Prénom				
2	*L*				
5	**Prénom**	**Date**	**Enfant**	**Logement**	**Voiture**
7	Magali	1/07/2011	2	Loft	Toyota
8	Isabelle	10/05/2011		Studio	Volvo
10	Alain	15/04/2012	1	Studio	Ford
11	Philippe	23/06/2011		Studio	
14	Julien	11/07/2011	2	Maison	Mercedes
16	Aloys	4/03/2012	1	Maison	Volkswagen

Cherchons les voitures dont la première lettre est inconnue, les deuxième et troisième lettres **OL** et quels que soient le nombre et la valeur des caractères qui suivent.

	Voiture				
2	?OL*				
5	**Prénom**	**Date**	**Enfant**	**Logement**	**Voiture**
8	Isabelle	10/05/2011		Studio	Volvo
13	Eric	14/05/2011		Studio	Volvo
16	Aloys	4/03/2012	1	Maison	Volkswagen

9.2. Les champs vides et non vides

Dans la cellule contenant le filtre, saisissez :

- = ou "=" pour les enregistrements non remplis ;
- <> ou "<>" pour les enregistrements remplis**.**

Filtrons les personnes qui n'ont pas de voiture.

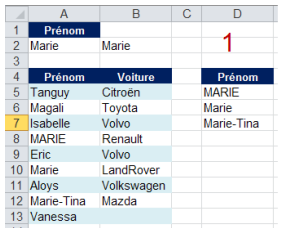
	Voiture				
2	=				
5	**Prénom**	**Date**	**Enfant**	**Logement**	**Voiture**
9	Stéphane	12/12/2011	3	Loft	
11	Philippe	23/06/2011		Studio	
18	Vanessa	19/03/2012	3	Appartement	
20	Sandra	28/12/2012	1	Studio	

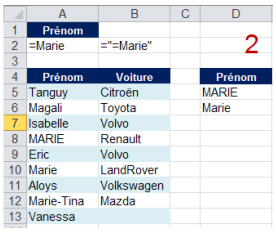
A contrario, filtrons maintenant ceux qui ont une voiture.

	Voiture				
2	<>				
5	**Prénom**	**Date**	**Enfant**	**Logement**	**Voiture**
6	Tanguy	8/01/2012		Studio	Citroën
7	Magali	1/07/2011	2	Loft	Toyota
8	Isabelle	10/05/2011		Studio	Volvo
10	Alain	15/04/2012	1	Studio	Ford
12	Marie	18/08/2011	1	Loft	Renault
13	Eric	14/05/2011		Studio	Volvo
14	Julien	11/07/2011	2	Maison	Mercedes
15	Sandra	25/10/2011	3	Maison	LandRover
16	Aloys	4/03/2012	1	Maison	Volkswagen
17	Marie-Tina	3/11/2012	2	Studio	Mazda
19	Benedicte	15/04/2011	2	Appartement	Peugeot

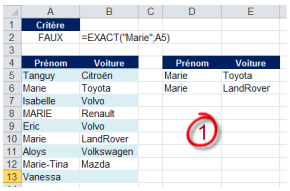
10. À savoir

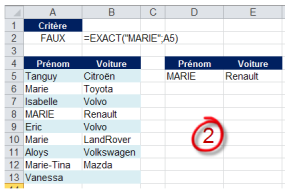
Si on veut filtrer une chaîne exacte par exemple **Marie**, il y a lieu de faire précéder cette chaîne par l'apostrophe et l'opérateur logique d'égalité **'=Marie** ou **="=Marie"** (exemple 2) afin de ne pas filtrer **Marie-Tina** comme le montre l'exemple 1.





Les filtres sur les chaînes de caractères ne sont pas sensibles à la casse. Pour pallier le problème, il y a lieu d'utiliser un critère calculé.





11. En conclusion, nous retiendrons

Les données peuvent être filtrées sur place ou exportées vers un autre emplacement.
Les étiquettes de la zone de critères doivent être identiques à celles de la table de données à l'exception des critères calculés.
Les critères placés sur une même ligne sont équivalents à un **ET**, les critères placés sur une deuxième ligne à un **OU**.
Ne pas englober une ligne vide dans la plage des critères.
Les étiquettes des critères calculés **DOIVENT** avoir un nom différent des étiquettes de la table de données.
Les critères sont des formules dont le résultat doit être **VRAI** ou **FAUX** (booléennes) et doivent concerner la première cellule de la colonne concernée.
Pour indiquer qu'on veut copier les données filtrées vers un autre emplacement, il faut cocher l'option Copier vers un autre emplacement et remplir la zone Copier dans avec :

- l'adresse de la première cellule de l'emplacement désiré si l'on exporte tous les champs (ex : **$M$1**) ;
- le nom des étiquettes des colonnes à exporter (avec la même orthographe !) et sélectionner ces étiquettes (ex : **$M$1:$Q$1**) ; les étiquettes peuvent être placées dans un ordre différent des étiquettes de la table de données.

Si on veut exporter vers une autre feuille, il faut procéder à l'envers : il faut exécuter le filtre à partir de la feuille de destination.

11.1. Ce qu'il faut savoir

Dans la zone des critères, ne pas référencer la troisième ligne (la ligne OU) si celle-ci ne contient rien. Il en résulterait un filtrage erroné.
Se méfier des propositions faites par la boîte de dialogue. Il arrive que celle-ci vous propose des références erronées.
En effet l'outil **Filtre élaboré** crée des références nommées pour les options plage, exportation et critères qu'il utilise pour faire des propositions par l'intermédiaire de sa boîte de dialogue.

12. VBA et les filtres élaborés

L'utilisation des filtres élaborés en VBA est gérée par la méthode **AdvancedFilter** de l'objet **Range**.
Cette méthode filtre ou copie des données à partir d'une liste basée sur une plage de critères.

12.1. Syntaxe

expression . **AdvancedFilter** (Action, [CriteriaRange], [CopyToRange], [Unique])
Les arguments : à l'exception d'Action, tous les arguments sont facultatifs.
Action : Deux constantes de type **xlFilterAction** définissent le type d'action.

Nom	Valeur	Description
xlFilterCopy	2	Copie les données filtrées vers un nouvel emplacement
xlFilterInPlace	1	Filtre les données en place

CriteriaRange Zone de critères.
CopyToRangePlage de destination des lignes copiées si l'argument Action a comme valeur **xlFilterCopy.**
Unique**True** pour filtrer exclusivement les enregistrements uniques et **False** pour filtrer tous les enregistrements répondant aux critères. La valeur par défaut est False.

12.2. En pratique

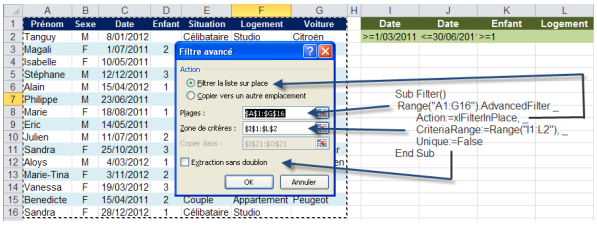
Voyons quelques exemples de l'utilisation du code VBA avec le **Filtre avancé**.

12.3. Premier exemple - Filtrer la liste sur place

Pour notre premier exemple, filtrons sur place la plage **A1G16**, selon les critères suivants : les dates comprises entre le 1/3/2011 et le 30/06/2011 et ayant des enfants.
Appelons cette procédure **Filter**.

```
Sub Filter()
  Range("A1:G16").AdvancedFilter _
          Action:=xlFilterInPlace, _
          CriteriaRange:=Range("I1:L2"), _
          Unique:=False
End Sub
```

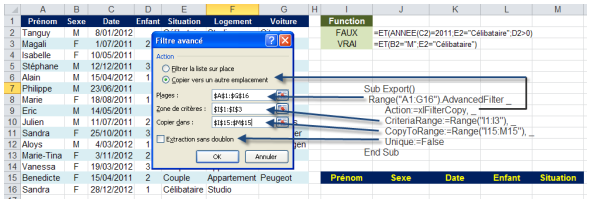
L'illustration ci-dessous montre les arguments de la méthode **AdvancedFilter** en regard de la boîte de dialogue **Filtre avancé.**



12.4. Deuxième exemple – Exporter la liste filtrée

Pour cet exemple, filtrons en copiant vers un autre emplacement.
Cette fois-ci nous emploierons comme critères deux cellules contenant chacune la fonction **ET**.
La procédure est nommée **Export**.

```
Sub Export()
  Range("A1:G16").AdvancedFilter _
          Action:=xlFilterCopy, _
          CriteriaRange:=Range("I1:I3"), _
          CopyToRange:=Range("I15:M15"), _
          Unique:=False
End Sub
```



13. Sky is the limit

Cette expression s'adapte bien à ce que l'on peut faire avec VBA quand on maîtrise bien Excel.
La feuille [**ControlsFilters**] illustre les innombrables possibilités qui s'offrent à toutes personnes voulant développer rapidement des exportations de tables de données sans modifier le code VBA, en utilisant les cellules d'Excel comme paramètres.
Pour profiter pleinement de cette procédure afin de la rendre utilisable quelle que soit la situation, il y a lieu de créer une procédure.
Les exemples présents dans cette feuille utilisent une fonction nommée **ExportByFilter** qui permet de passer comme arguments la table de données, la zone des critères et en option la référence à la zone d'export. Cette procédure renvoie le nombre de données filtrées et donc exportées.

```
Function ExportByFilter(znData As Range, znCriteria As Range, Optional znExport As Range) As Long
 ' Procédure d'exportation basée sur le filtre élaboré
 ' Author : Philippe Tulliez http://philippe.tulliez.be
 ' Valeur renvoyée : Nombre d'enregistrements exportés
 ' znData     ' Table de données
 ' znCriteria ' Zone des critères
 ' [znExport]   ' Zone d'exportation (si vide exporte tout en créant une feuille)
 If znExport Is Nothing Then ' Création de la feuille d'export et coloration en rouge
  Worksheets.Add before:=Sheets(1)
  With Worksheets(1): ActiveCell = .Range("A1") : .Tab.Color = vbRed: End With
  Set znExport = ActiveCell
 End If
 znData.AdvancedFilter Action:=xlFilterCopy, CriteriaRange:=znCriteria, CopyToRange:=znExport
 ExportByFilter = znExport.CurrentRegion.Rows.Count &#150; 1
End Function
```

Cette procédure est présente dans le module **mTutoAdvancedFilter** qui accompagne le tutoriel.
Pour tester la création automatique de la feuille d'exportation, il suffit d'effacer une des données dans les cellules contenant les noms des plages d'exportation.

14. Fichier exemple

Ce tutoriel est accompagné d'un fichier exemple que vous pouvez télécharger en cliquant ici : Lien 01

Retrouvez l'article de Philippe Tulliez en ligne : Lien 02

Application MS ACCESS dans le Cloud Storage

Comment utiliser une solution de Cloud Storage avec Microsoft ACCESS ? Quels sont les avantages, les inconvénients et surtout comment les mettre en œuvre ?

1. Introduction

Dans ce tutoriel, nous allons aborder l'utilisation d'une application Microsoft ACCESS sur un Cloud Storage.

1.1. Avertissement

Le Cloud Storage, du fait de son fonctionnement par synchronisation de fichiers, ne permet pas les accès concurrents.
Pour aborder ce tutoriel, vous devez avoir de sérieuses connaissances en VBA et maîtriser l'utilisation des classes. Vous devez également maîtriser les tables liées, ainsi que la gestion des conflits.
Retrouvez un excellent tutoriel pour vous former aux classes VBA : Lien 03.

2. Prérequis

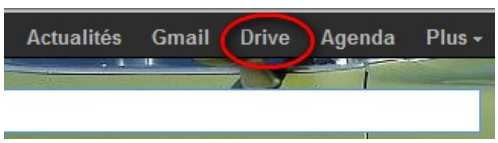
Outre la possession d'une connexion haut débit permanente, vous devrez installer une des solutions de Cloud Storage et vous documenter sur sa gestion de conflits pour adapter la classe.
La solution choisie pour illustrer ce tutoriel est **Dropbox** dont un lien est disponible ci-dessous. La classe est adaptée à ce produit, libre à vous de l'adapter pour un autre.
Les solutions de Cloud Storage possèdent toutes une offre gratuite, mais peuvent être plus ou moins avantageuses en termes techniques (possibilités, capacité de stockage, fonctionnement, rapidité, gestion des conflits).

2.1. Skydrive de Microsoft

Vous ne pouvez l'installer qu'à partir de **Windows Vista** en version 32 ou 64 bits. Plus d'informations sur cette page : Lien 04

2.2. GDrive de Google

Installable à partir de XP, il dispose d'une capacité de stockage de 5 Go gratuit.
Pour l'installer, rendez-vous sur votre **iGoogle**. **GDrive** est présent dans la barre de menu.



2.3. Dropbox de Dropbox Inc.

Dropbox propose 2 Go d'espace de stockage extensible. J'ai choisi cette solution pour illustrer ce cours.

Vous pouvez faire l'installation dès à présent en utilisant l'adresse suivante : Lien 05 (gagnez 500 Mo supplémentaires en utilisant ce lien de parrainage)

3. Fonctionnement

Ces solutions de stockage fonctionnent toutes sur le mode de la synchronisation de fichiers.
Un plug-in, petit logiciel résident, est installé sur le poste en même temps qu'un répertoire est créé. Lorsque l'utilisateur copie, crée, modifie ou supprime un fichier, le changement est automatiquement répercuté sur le Cloud. Enfin les serveurs renvoient le changement vers les autres postes abonnés.
Les postes ne sont donc pas connectés à un espace de travail sur un serveur. Il n'y a pas à proprement parler un espace de travail sur un serveur, juste un stockage sur une multitude de serveurs qui ne sont pas identifiables par l'utilisateur.

3.1. Schéma de fonctionnement du Cloud Storage

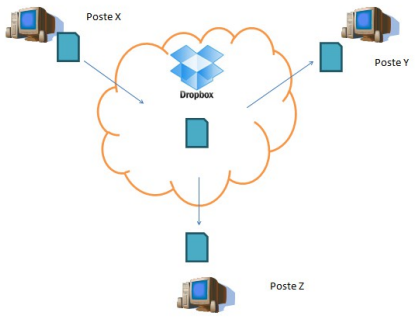
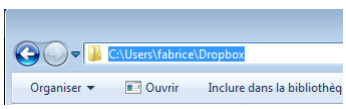


Schéma de fonctionnement du Cloud Storage

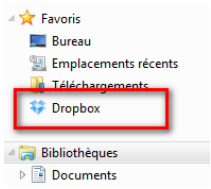
3.2. Installation

À partir du lien indiqué plus haut, vous avez installé **Dropbox** sur votre poste. Le plug-in est chargé au démarrage de la session et on peut le contrôler via la barre des tâches.

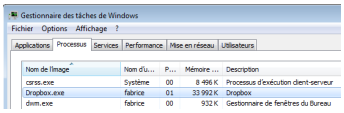
Lors de l'installation, le dossier est créé par défaut dans le dossier de l'utilisateur.



Un raccourci vers ce dossier est également créé sur le bureau.



Le plug-in reste en mémoire jusqu'à sa fermeture et est accessible dans la barre des tâches. C'est lui qui observe les fichiers et les synchronise.



Notez qu'une liaison internet haut débit est obligatoire pour avoir un fonctionnement optimal.

4. Le Cloud appliqué à MS ACCESS

Outre la gratuité, le partage de données d'une application est un avantage indéniable. Il convient cependant d'être objectif, ce n'est pas la solution miracle et ça ne remplacera pas une base de données en ligne du type **Microsoft Azure**, notamment pour les accès concurrents.

4.1. Inconvénient

Le seul inconvénient majeur et sûrement rédhibitoire pour certaines applications est de ne pas pouvoir l'appliquer à un fonctionnement multiutilisateur. En effet, comme il s'agit de synchronisation de fichiers, si plusieurs machines modifient le même fichier de données en même temps, il y aura conflit.
En cas de conflit, les fichiers ne sont pas perdus ; leur nom est complété par le nom de la machine et la date du conflit.
Voici un exemple de conflit sur deux fichiers.

4.2. Avantages

Outre la gestion des conflits indiquée précédemment, un avantage non négligeable est la taille des fichiers. **Dropbox** bénéficie d'une marge confortable avec ses 10 Go maxi par fichier, soit bien au-delà d'un fichier **Microsoft Access**.
Vous disposez également d'un archivage des versions sur une durée de trente jours (extensible moyennant un abonnement payant) et également la possibilité de partager des répertoires individuellement.

4.3. Précautions

Il convient de choisir une architecture frontale/dorsale pour éviter de synchroniser des informations inutiles, comme l'IHM.
N'hésitez pas à crypter la dorsale, **Dropbox Inc.** vous l'autorise.

5. Fonctionnement de l'application

Avec une architecture Frontale/Dorsale, **Microsoft ACCESS** n'accède au fichier dorsal que lorsqu'il y a accès à l'une des tables. Si le fichier dorsal est ouvert, il sera forcément synchronisé à sa fermeture. C'est en gardant à l'esprit ce mécanisme que nous allons travailler.
Vous l'aurez compris, cette technique proscrit l'utilisation des accès concurrents. Il faut donc se prémunir contre tout risque d'accès multiples en s'assurant qu'au démarrage de l'application personne n'est connecté et ainsi éviter la génération de conflits.

5.1. Comment ça marche

Dès le démarrage et avant même tout accès à une des tables de la dorsale, l'application vérifie qu'un fichier, que nous appellerons « lock », existe dans le dossier synchronisé.

- Si ce n'est pas le cas, l'application le crée. Dès lors, le plug-in **Dropbox** l'envoie dans le Cloud, qui le renvoie vers les postes abonnés. L'application fonctionne normalement. À la fermeture, le fichier « lock » est effacé et le plug-in répercute l'action dans le Cloud.
- Si le fichier « lock » existe, l'application affiche un message et se ferme.

5.1.1. Schéma de fonctionnement

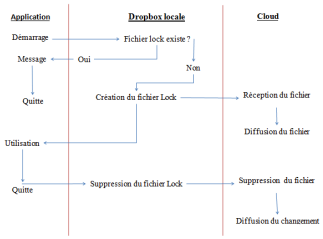


Schéma de fonctionnement

5.2. Fichier lock vs laccdb

Pourquoi utiliser un fichier dit « lock » au lieu d'utiliser directement le fichier « laccdb » de la dorsale ?
La question mérite d'être posée. Le choix s'est porté sur un fichier texte pour pouvoir contrôler les homonymies de nom d'utilisateur.

6. Mise en pratique

Il est quand même plus facile de gérer ce genre de fonctionnalité dans une entité. Les classes feront donc pleinement l'affaire.

6.1. Table de paramètres

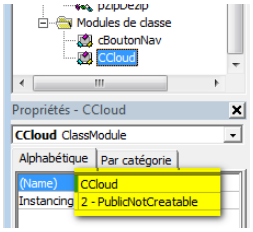
Comme nous l'avons vu dans l'installation, le chemin **Dropbox** local comme l'utilisateur peuvent être différents d'un poste à l'autre. Une table locale à l'application est donc nécessaire pour stocker ces deux paramètres importants.
Créez une table dans l'application suivant cette structure :

Nom	Type	Longueur
Idiparametre	NumeroAuto	
Parametre	Texte	25
Valeur	Texte	255

Parametre contiendra le nom du paramètre et **Valeur** sa valeur. Cette table contiendra le paramètre **CheminPartage** qui donne le chemin vers la **Dropbox** locale et **Utilisateurlocal,** l'utilisateur local de l'application.
Inutile de la remplir, c'est le code qui s'en chargera.
Enregistrez-la sous le nom **tiParametres**. Le i indique que cette table est interne à l'application (fichier Frontal), par opposition à une table de paramètres qui serait stockée dans la dorsale.

6.2. Création de la Classe CCloud

Créez un module de classe nommé **CCloud**, n'oubliez pas de le paramétrer grâce aux propriétés comme l'indique la copie d'écran suivante.



Nous sommes fin prêts pour créer les attributs et les méthodes.

6.3. Attributs et méthodes

Voici le diagramme de classe **CCloud** qui permet d'en avoir une vue d'ensemble.

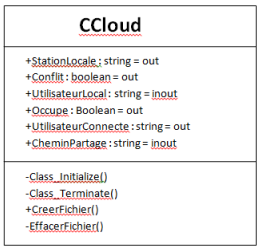


Schéma de classe

6.3.1. Attributs

Nom	Description
StationLocale	Le nom de la station locale
Conflit	Indique s'il y a conflit de fichier dorsal
UtilisateurLocal	L'identifiant CCloud de l'utilisateur local
Occupe	La détection de l'utilisation de l'application par un tiers
UtilisateurConnecte	L'identifiant CCloud de l'utilisateur connecté
CheminPartage	Le chemin local de la dorsale (Dossier Dropbox)

6.3.2. Méthodes

Nom	Description
Class_Initialize	Initialisation de la classe
Class_Terminate	Terminaison de la classe
CreerFichier	Création du fichier « lock »
EffacerFichier	Effacement du fichier « lock »

6.4. En-tête de classe

L'en-tête est important dans une classe, car c'est là qu'on y retrouve les variables locales.

```
Option Compare Database
Option Explicit

Dim vStationLocale As String        ' nom de la
station locale
Dim vCheminPartage As String        ' le chemin
SkyDrive/DropBox/GDrive local
Dim vUtilisateurLocal As String     '
l'utilisateur local
Const cNomFichier As String = "occupe.txt"  'nom
du fichier lock
```

La constante **cNomFichier** contient le nom du fichier « Lock ». Si ce nom ne vous convient pas, vous pouvez très bien le modifier ; le fonctionnement n'en souffrira pas.

6.5. Initialize/Terminate

La méthode d'initialisation **Class_Initialize** contient l'initialisation de certaines variables comme le nom de l'utilisateur local ainsi que celui de la station.
La méthode **Class_Terminate,** qui s'exécute à la libération de la classe, permet de déclencher la méthode d'effacement du fichier « lock » et de contrôler un éventuel conflit.

```
Private Sub Class_Initialize()
' démarre la classe
  vStationLocale = Environ("ComputerName")
  'on fixe le nom de l'utilisateur si celui-ci
n'existe pas.
  If vUtilisateurLocal = "" Then
     UtilisateurLocal = vStationLocale &amp;amp; ";"
&amp;amp; Environ("UserName")
  End If
End Sub

Private Sub Class_Terminate()
'Fermeture de la classe
'à appeler sur le close du dernier formulaire
ouvert ou avant un quit
' supprime le fichier lock
    EffacerFichier
End Sub
```

6.6. Les attributs

Certains attributs n'ont pas de Setter car ils renvoient des valeurs initialisées par la classe.

6.6.1. Le nom de la station locale - StationLocale

Le nom de la station sera utile pour déterminer un conflit d'écriture sur la dorsale au moment de la fermeture.

```
Public Property Get StationLocale() As String
    StationLocale = vStationLocale
End Property
```

Cet attribut est renseigné au moment de l'initialisation de

la classe par la méthode **Class_Initialize**. Il renvoie le nom de la station locale.

6.6.2. L'utilisateur local - UtilisateurLocal

Pour cet attribut vous devez créer le Getter et le Setter.

Le **Set** n'est utilisé que pour un type **Objet**, dans le cas d'un type natif (string, long, boolean...) , il faut utiliser **Let**.

```
Public Property Let UtilisateurLocal(vLocalUser
As String)
' définit l'utilisateur local à partir de
' nom d'utilisateur Windows +
On Error GoTo errSub

Dim db As DAO.Database
Dim rst As DAO.Recordset
Set db = CurrentDb
Set rst = db.OpenRecordset("tIParametres",
dbOpenDynaset)
rst.FindFirst "Parametre='UtilisateurLocal'"
' le paramètre existe on le renseigne
If Not rst.NoMatch Then
   rst.Edit
   rst.Fields(2) = vLocalUser &amp; "-" &amp;
Replace(CDbl(Now()), ".", "")
   rst.Update
Else
' il n'existe pas on le crée
   rst.AddNew
   rst.Fields(1) = "UtilisateurLocal"
   rst.Fields(2) = vLocalUser &amp; "-" &amp;
Replace(CDbl(Now()), ".", "")
   rst.Update
End If

rst.Close

ExitSub:
  Set rst = Nothing
  Set db = Nothing
  On Error GoTo 0
  Exit Property

errSub:
  Resume Next
End Property
```

Le Setter de cet attribut stocke le nom de la station et de l'utilisateur. Pour rendre sa valeur unique, nous utilisons l'astuce d'ajouter **la date/heure/minute/seconde** au format **Double**, son format natif de stockage, auquel nous enlevons le séparateur décimal.
En voici un exemple :

```
PC-WINDOWS7;fabrice;412348217708333
```

Ce nouveau nom d'utilisateur unique sera inscrit dans le fichier « lock » où l'application ira le contrôler.
Le Getter est construit sur le même modèle.

```
Public Property Get UtilisateurLocal() As String
' renvoie l'utilisateur local si défini
On Error GoTo errSub
' renvoie la valeur de l'attribut demandé
Dim db As DAO.Database
Dim rst As DAO.Recordset

Set db = CurrentDb
Set rst = db.OpenRecordset("tIParametres",
dbOpenSnapshot)
rst.FindFirst "Parametre='UtilisateurLocal'"
If Not rst.NoMatch Then
   UtilisateurLocal = Nz(rst.Fields("Valeur"),
"")
Else
   UtilisateurLocal = ""
End If
rst.Close
ExitSub:
  Set rst = Nothing
  Set db = Nothing
  On Error GoTo 0
  Exit Property

errSub:
    GoTo ExitSub
End Property
End Property
```

Attention ! Dans le cas où il n'y pas d'utilisateur enregistré, ce qui ne devrait pas se produire, on utilise NZ() pour éviter une erreur de typage au retour de la valeur.

6.6.3. Le chemin local de la Dropbox - CheminPartage

Pour le chemin de la **Dropbox**, le répertoire contenant la dorsale, nous allons utiliser la même méthode que précédemment.

```
Public Property Let CheminPartage(vChemin As
String)
' enregistre le chemin de partage
On Error GoTo errSub

Dim db As DAO.Database
Dim rst As DAO.Recordset
Set db = CurrentDb
Set rst = db.OpenRecordset("tIParametres",
dbOpenDynaset)
rst.FindFirst "Parametre='CheminPartage'"
If Not rst.NoMatch Then
   rst.Edit ' enregistre le chemin de partage
   rst.Fields(2) = vChemin
   rst.Update
Else
   rst.AddNew  ' crée le paramètre
   rst.Fields(1) = "CheminPartage"
   rst.Fields(2) = vChemin
   rst.Update
End If
' stockage du chemin pour la classe
vCheminPartage = vChemin
rst.Close
ExitSub:
  Set rst = Nothing
  Set db = Nothing
  On Error GoTo 0
  Exit Property

errSub:
  Resume Next
End Property
```

Le Getter est similaire à la différence près que s'il est

connu, on ne va pas le chercher.

```
Public Property Get CheminPartage() As String
' renvoie le chemin de partage si déjà défini
On Error GoTo errSub

If Len(vCheminPartage) <> 0 Then
   CheminPartage = vCheminPartage
   Exit Property
End If

Dim db As DAO.Database
Dim rst As DAO.Recordset
Set db = CurrentDb
Set rst = db.OpenRecordset("tIParametres",
dbOpenSnapshot)
rst.FindFirst "Parametre='CheminPartage'"
If Not rst.NoMatch Then
   CheminPartage = Nz(rst.Fields("Valeur"), "")
Else
   CheminPartage = ""
End If

rst.Close

ExitSub:
  Set rst = Nothing
  Set db = Nothing
  On Error GoTo 0
  Exit Property

errSub:
    GoTo ExitSub
End Property
```

6.6.4. L'utilisateur connecté - UtilisateurConnecte

Il est inutile de stocker l'utilisateur connecté. Un Getter suffira largement pour cet attribut.
Pour le connaître, on ouvre le fichier « lock » et on lit la seule ligne présente.

```
Public Property Get UtilisateurConnecte() As
String
' retourne l'utilisateur qui est connecté
actuellement
' celui-ci n'est pas forcément l'utilisateur
local
On Error Resume Next ' une erreur peut se
produire si le fichier lock
                     ' est fermé dans
l'intervalle
    Dim vFile As Long
    Dim strLigne As String
    Dim strUtilisateur() As String

    If Dir(vCheminPartage &amp; cNomFichier,
vbNormal) <> "" Then
        vFile = FreeFile
        Open vCheminPartage &amp; cNomFichier For
Input As vFile
        Line Input #vFile, strLigne
        Close #vFile        ' Ferme le fichier.
    End If

    strUtilisateur = Split(strLigne, ";")
    UtilisateurConnecte = strUtilisateur(0) &amp;
";" &amp; strUtilisateur(1)

End Property
```

Comme l'utilisateur local, le connecté est composé du **nom de la station**, du **nom de l'utilisateur Windows** et du **numéro date/heure/minute/seconde** sans caractère décimal.
Nous ne prenons que les deux premiers paramètres soit le nom de la station et de l'utilisateur. Notez que vous pouvez aller plus loin en testant la durée de la connexion. Cela peut vous renseigner sur un éventuel problème de libération de l'application.

6.6.5. La détection de l'utilisation de l'application par un tiers - Occupe

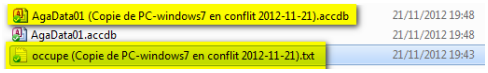
Cet attribut ne possède qu'un Getter. En effet, il n'y pas de stockage possible puisque c'est la photographie à un instant T de l'état de l'application.

```
Public Property Get Occupe() As Boolean
' Renvoie vrai si le fichier de lock existe
' et que l'utilisateur n'est pas le local
Dim vOccupe As Boolean
    vOccupe = Dir(vCheminPartage &amp;
cNomFichier, vbNormal) <> ""
    If vOccupe Then
       If UtilisateurConnecte = UtilisateurLocal
Then
          vOccupe = False ' c'est la même
connexion
       End If
    End If
    Occupe = vOccupe
End Property
```

Pour déterminer si l'application est occupée, on teste l'existence du fichier « lock » et si l'utilisateur connecté est différent de l'utilisateur local.

6.6.6. La détection des conflits - Conflit

Pour détecter un conflit, il faut tester l'existence d'une copie de la base effectuée par le serveur Dropbox.
La copie se nommera ainsi :



Nom de la dorsale (copie de **Nom de la station** en conflit **Date au format anglo-saxon**).accdb

Un **Dir** donnera ceci :

```
Dir *Pc-windows7 en conflit 2012-11-21.accdb
```

En voici l'équivalent avec l'attribut de la classe.

```
Public Property Get Conflit() As Boolean
On Error GoTo errSub
Conflit = Dir(CDropb.CheminPartage &amp; "*"
&amp; CDropb.StationLocale &amp; " en conflit "
&amp; Format(Date, "yyyy-mm-dd") &amp; ").accdb")
<> ""
ExitSub:
  On Error GoTo 0
  Exit Property
errSub:
    GoTo ExitSub
```

```
End Property
```

Cet attribut est appelé depuis un formulaire muni d'un timer.

6.7. Les méthodes

Il n'y a que deux méthodes à mettre en place pour parfaire la classe.

- Création du fichier « lock ».
- Effacement du fichier « lock ».

Rien ne vous empêche de l'étoffer suivant vos besoins personnels.

6.7.1. Création du fichier - CreerFichier

On crée le fichier « lock » simplement à partir d'un Open. On y stocke l'utilisateur local et la date/heure de connexion.

```
Public Sub CreerFichier()
' création du fichier de lock
' composé de l'utilisateur local + date heure de connexion
    Dim vFile As Long
    Dim strLigne As String
    vFile = FreeFile
    strLigne = UtilisateurLocal &amp; ";" &amp; Format(Now(), "dddd dd mmmm yyyy hh:mm:ss")

    Open vCheminPartage &amp; cNomFichier For Append As vFile
    Print #vFile, strLigne
    Close #vFile        ' Ferme le fichier.

End Sub
```

6.7.2. Effacement du fichier Lock - EffacerFichier

L'effacement comporte une condition. Les utilisateurs, le local et le connecté doivent être les mêmes pour pouvoir supprimer le fichier.

```
Private Sub EffacerFichier()
' efface le fichier (lors de la libération)
' uniquement si l'utilisateur actuel est l'utilisateur local
    If UtilisateurConnecte = UtilisateurLocal Then
       Kill vCheminPartage &amp; cNomFichier
    End If
End Sub
```

Cette méthode est appelée lorsque la classe est déchargée.

7. Mise en œuvre de la classe CCloud

Tout d'abord il faut la déclarer dans l'en-tête de module pour qu'elle vive tout le temps de l'activité de l'application.

```
Option Compare Database
Option Explicit
Public CDropb As New CCloud
```

Ensuite, dans une procédure appelée au démarrage, on lance la procédure d'attachement automatique qui retourne le chemin.

```
    CDropb.CheminPartage = pGestionTables.fgAttache  ' attache les tables et renvoie le chemin de la Dorsale
```

Dans la procédure d'attachement, les variables globales sont :

- strLstTable contient la liste des tables à attacher ;
- strFichierData contient le nom de la dorsale ;
- strPassWordBD contient le mot de passe pour l'attache.

```
Global Const strlstTable As String = "tTable1;tTable2;tTable3"
' nom du fichier de base de données à attacher
Global Const strFichierData As String = "AgaData01.accdb"
Global Const strPassWordBD As String = "monmotdepasse"

Function fgAttache() As String
'-----------------------------------------------
' Procedure : pgAttache
' Author    : Fabrice CONSTANS (MVP)
' Date      : 18 / 07 / 2012
' Purpose   : Procédure standard d'attachement des tables
'               Vérifie l'attachement ou le refait
' Parametres: Utilise strlstTable qui contient les tables
'                   strFichierData qui contient le fichier des tables
' Return    : chemin de la dorsale
'-----------------------------------------------
'
    On Error GoTo err_demarrage

    Dim db As DAO.Database
    Dim rst As Recordset        ' pour le test d'attache
    Dim lstTable() As String    ' contient les tables
    Dim i As Integer            ' the Compteur
    Dim tbl As TableDef
    Dim strChemin As String

    ' utilise le chemin de la table des paramètres
    strChemin = CDropb.CheminPartage &amp; strFichierData
    ' si le fichier Dorsal n'existe pas on le demande
    If Not FichierExiste(strChemin) Then
        strChemin = fIndiqueFichier(strChemin)
    Else
        strChemin = strChemin
    End If

    ' sauve le chemin dans la table des paramètres internes
    CDropb.CheminPartage = fFichierExt(strChemin, eChemin)
    ' si user local n'est pas user courant
    If CDropb.Occupe Then
        MsgBox "La base est actuellement utilisée par : " &amp; CDropb.UtilisateurConnecte &amp; vbCrLf &amp; "Faites une tentative ultérieurement.", vbExclamation + vbOKOnly, fVersionProduit()
        ' libération de la classe
```

```
        Set CDropb = Nothing
        DoCmd.Quit  'quitte l'application
    End If
    ' Création du fichier, réservation de
l'application
    CDropb.CreerFichier

    ' à partir de là on commence les procédures
d'attachement sur la dorsale
    lstTable = Split(strlstTable, ";")      ' la
liste des tables à traiter

    Set db = DBEngine(0)(0)
    For i = 0 To UBound(lstTable)           '
liste les tables et tente l'ouverture
        Set rst = db.OpenRecordset(lstTable(i),
dbOpenSnapshot)
        rst.Close
        Set rst = Nothing
    Next

    db.TableDefs.Refresh
    If strChemin = "" Then
        strChemin =
db.TableDefs(lstTable(0)).Connect
        strChemin = Right(strChemin,
Len(strChemin) - InStr(1, strChemin,
";DATABASE=") - 9)
    End If
    fgAttache = fFichierExt(strChemin, eChemin)
    Exit Function

err_demarrage:

    If Err.Number = 3078 Then  ' ne trouve pas la
table
        Set tbl = db.CreateTableDef(lstTable(i))
        tbl.Connect = "MS Access;PWD=" &amp;
strPassWordBD &amp; ";DATABASE=" &amp; strChemin
'Me.CheminBD '";DATABASE=" &amp;
        tbl.SourceTableName = lstTable(i)
        db.TableDefs.Append tbl
        db.TableDefs(tbl.Name).RefreshLink
        Resume
    End If

    For Each tbl In db.TableDefs
        If tbl.Attributes = dbAttachedTable Then
            tbl.Connect = "MS Access;PWD=" &amp;
strPassWordBD &amp; ";DATABASE=" &amp; strChemin
            db.TableDefs(tbl.Name).RefreshLink
        End If
    Next
    Resume
End Function
```

Il y a plusieurs lignes faisant référence à notre classe.

- Récupération du chemin de la Dropbox et du nom de la dorsale.

```
    strChemin = CDropb.CheminPartage &amp;
strFichierData
```

- Si l'application est utilisée par un tiers. On affiche le message d'avertissement, on libère la classe et on quitte l'application.

```
    If CDropb.Occupe Then
        MsgBox "La base est actuellement utilisée
par : " &amp; CDropb.UtilisateurConnecte &amp;
vbCrLf &amp; "Faites une tentative
ultérieurement.", vbExclamation + vbOKOnly,
fVersionProduit()
        ' libération de la classe
        Set CDropb = Nothing
        DoCmd.Quit  'quitte l'application
    End If
```

- On crée le fichier « lock ».

```
    CDropb.CreerFichier
```

L'application peut poursuivre son activité.

7.1. Procédure à la fermeture de l'application

À la fermeture de l'application, il faut libérer l'application en supprimant le fichier « lock ».

7.1.1. Événement Sur Fermeture

Sur cet événement, on ouvre un formulaire d'attente et de contrôle des conflits. Le mode **acDialog** stoppe l'exécution jusqu'à la fermeture du formulaire.

```
Private Sub Form_Close()
    DoCmd.OpenForm "fAttente", , , , , acDialog
    Set CDropb = Nothing
    Doevents
    DoCmd.Quit
End Sub
```

8. Formulaire fAttente

Le formulaire **fAttente** permet de faire patienter l'utilisateur pendant que l'application vérifie, à intervalles réguliers, les conflits matérialisés par la présence d'une copie du fichier dorsal.
J'ai choisi cette représentation, libre à vous de faire la vôtre !

8.1. Les contrôles

Ce formulaire est composé des contrôles suivants :

- une cadre qui fait office de barre de progression ;
- une zone de titre ;
- une zone de message ;
- un bouton de commande.

Nom	Visible	Contenu	Type
Barre	oui	Couleur	Cadre
TxtMsg	non	Vide	Étiquette
btnFermer	non	Quitter	Bouton de commande
EtiqTitre	oui	Vérification des conflits éventuels	Étiquette

8.2. Le code

Nous avons besoin d'une variable locale pour ce module.

Elle est donc déclarée dans l'en-tête du module. Cette variable permettra de faire évoluer la barre de progression.

```
Option Compare Database
Option Explicit
Dim i As Long 'stocke le pas de progression pour la barre
```

8.2.1. Sur Ouverture

Sur l'événement d'ouverture, nous initialisons la variable à 500 qui est la valeur du pas de progression de la barre.

```
Private Sub Form_Open(Cancel As Integer)
i = 500 'pas de progression
End Sub
```

8.2.2. Le bouton Fermer

Il s'agit du bouton qui permettra de fermer le formulaire en cas de détection de conflit et d'affichage du message.

```
Private Sub btnFermer_Click()
    DoCmd.Close
End Sub
```

8.2.3. Le Timer

C'est le cœur du formulaire. C'est cet événement qui, déclenché à intervalles réguliers, testera s'il y a conflit et affichera le message.

```
Private Sub Form_Timer()

If Me.Barre.Width > i Then ' si la barre n'a pas atteint son dernier pas
    If Not CDropb.Conflit Then  ' si pas de conflit
       Me.Barre.Width = Me.Barre.Width - i 'progression de la barre
    Else
       ' si conflit
       Me.TxtMsg.Caption = "Vous avez créé un conflit d'écriture avec un autre utilisateur."
       Me.btnFermer.Visible = True 'on affiche le bouton
       Me.TxtMsg.Visible = True    ' on affiche le message
       Me.Barre.Visible = False    ' on cache la barre pour
       Me.TimerInterval = 0        ' on arrête le timer
    End If
Else
    DoCmd.Close ' on ferme automatiquement le formulaire si la progression est finie
End If

End Sub
```

9. Annexe

Dans ce chapitre, sont recensées les fonctions et procédures qui sont utilisées accessoirement dans la solution.

9.1. Fonction fHasBackSlash

Cette fonction renvoie la valeur passée avec un \ de terminaison.

```
Function fHasBackSlash(vlChemin As Variant) As String
'-----------------------------------------------
' Procedure : fHasBackSlash
' Author    : Fabrice CONSTANS
' Date      : 01 / 06 / 2012
' Purpose   : s'il n'y a pas de backslash à la fin d'un chemin passé en référence, l'ajoute
' Parametres:
' - vlchemin    = le chemin à modifier
'-----------------------------------------------
    On Error GoTo ExcfHasBackSlash
    ' teste si le caractère de la fin est un \
    If Right(Trim(vlChemin), 1) <> "\" Then
        ' l'ajoute
        fHasBackSlash = Trim(vlChemin) &amp;amp; "\"
    Else
        ' renvoie sans espace
        fHasBackSlash = Trim(vlChemin)
    End If
    On Error GoTo 0
    Exit Function

ExcfHasBackSlash:
    If Err.Number = 94 Then
        fHasBackSlash = "c:\"
        Exit Function
    End If

errSub:
    fHasBackSlash = False    ' une erreur quelconque se produit
End Function
```

9.2. Stockage documentaire

Vous pouvez indiquer le chemin de la Dropbox pour y stocker des documents liés à l'application.

```
vgStrRepertoireDocument = fHasBackSlash(CDropb.CheminPartage) &amp;amp; "Documents Amap\"
```

La fonction **fHasBackSlash()** teste et ajoute l'antislash en fin de chaîne. Elle est présente dans l'annexe en fin du tutoriel.

10. Conclusion

Comme vous l'avez constaté dans ce tutoriel, le Cloud Storage, malgré sa puissance, n'est pas adapté au fonctionnement des fichiers de données de Microsoft ACCESS lors d'accès concurrents. Ce que l'on économise financièrement, en faisant l'impasse sur une véritable solution de base de données hébergée, se paye en rigidité de fonctionnement et en développement de techniques de prévention.
Si vous souhaitez une vraie solution de base de données partagée, tournez-vous plutôt vers SQL Azure. Ce service est payant, mais vous n'aurez pas les inconvénients du Cloud Storage.

Retrouvez l'article de Fabrice Constans en ligne : Lien 06

Google met à jour l'API Google Maps qui permet désormais aux applications Android d'exploiter au maximum le service de cartographie

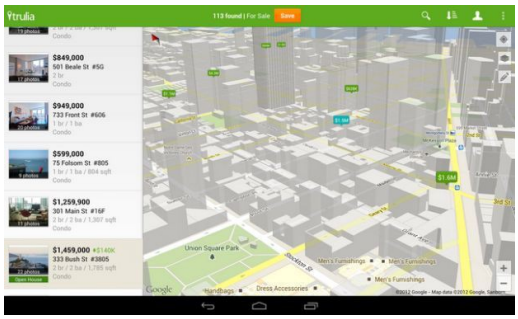
Google vient d'annoncer dans un billet de blog la mise à jour de son API Google Maps.

Pour rappel, Google Maps API est une bibliothèque permettant d'intégrer dans une application une carte du service Google Maps et d'utiliser ses fonctionnalités comme le zoom, le marker et bien d'autres.

Cette mise à jour de l'API concerne essentiellement les développeurs d'applications pour les terminaux Android.

Google Maps Android API v2 s'enrichit de plusieurs nouvelles fonctionnalités permettant d'exploiter au maximum le service Google Maps. Ceci, avec beaucoup plus de facilité.

L'API apporte la nouvelle fonction Android Fragments, qui permettra aux développeurs de créer des interfaces utilisateur plus dynamiques et souples pour les terminaux ayant des écrans de grande taille comme les tablettes.



Il est désormais possible d'ajouter plus de couches Google Maps comme les vues satellitaires, la gestion du trafic et les cartes d'intérieur pour de nombreux aéroports et des centres commerciaux dans les applications. Cette nouvelle version de la bibliothèque permet également de créer des marqueurs et des fenêtres d'information avec moins de code.

Le SDK Google Maps Android API s'installe comme une partie du SDK de l'OS mobile. La documentation d'utilisation de ses nouvelles fonctionnalités est disponible sur cette page : Lien 07. Une vidéo de présentation de ces nouveautés est également disponible.

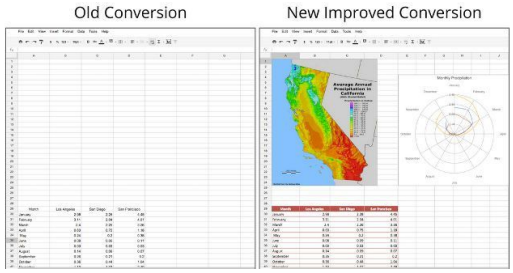
Voir la vidéo : Lien 08

Commentez la news de Hinault Romaric en ligne : Lien 09

Google Apps intègre Quickoffice pour s'attaquer à Microsoft Office qui fait évoluer CloudOn avec un meilleur support d'iOS et Android

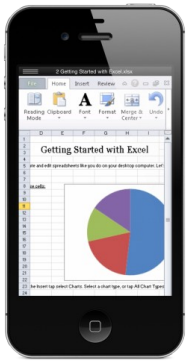
Google vient de procéder à une mise à jour de ses solutions de productivité et de bureautique Google Apps.

Cette évolution vise à séduire les utilisateurs de la suite bureautique de Microsoft via une meilleure intégration entre Google Apps et Quikoffice (NDLR : Suite bureautique acquise en juin dernier par Google). Les utilisateurs des outils de Google pourront ainsi convertir les fichiers Microsoft Office en documents Google et les éditer directement dans Quickoffice.



Quickoffice est actuellement disponible gratuitement sur l'iPad. Des versions pour smartphones sont également accessibles sur Android et iPhone.

En attendant la sortie d'une déclinaison d'Office pour iOS et Android, prévue pour 2013, Microsoft en réponse à Google, a procédé à une mise à jour de CloudOn, qui supporte désormais l'iPhone, l'iPad Mini et l'iPod Touch, et améliore le support d'Android et de l'iPad.



CloudOn 3.0 permet de créer et d'éditer des documents Word, Excel et PowerPoint directement à partir d'un mobile. Les documents peuvent être stockés dans le Cloud (SkyDrive, Dropbox, Google Drive) ou en local.

Commentez la news de Hinault Romaric en ligne : Lien 10

Le Wi-Fi sous Android

Nous allons voir dans ce tutoriel comment utiliser l'API WiFi d'Android.

1. Prérequis

Voici les prérequis pour ce tutoriel :

- projet Android 2.2 ;
- un téléphone/tablette ;
- un/plusieurs réseaux Wi-Fi ;
- avoir lu le tutoriel sur les listviews.

2. Permissions

Notre projet nécessite quelques permissions :

- android.permission.ACCESS_WIFI_STATE ;
- android.permission.CHANGE_WIFI_STATE.

La première sert à checker l'état du Wi-Fi, la seconde à changer celui-ci.

3. Ressources

Je vous donne directement les ressources nécessaires pour l'adapter sans les expliquer, c'est pour cela qu'il faut se référencer au tutoriel sur les listviews si jamais vous ne comprenez pas.
Le format des items pour notre listview :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
   android:orientation="horizontal"
android:layout_width="match_parent"
   android:layout_height="wrap_content">

   <TextView android:layout_height="fill_parent"
android:layout_width="10dp"
android:background="@android:color/white"
android:id="@+id/ForceSignal"/>

   <LinearLayout android:orientation="vertical"
       android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingLeft="10dp">

       <TextView android:id="@+id/tvWifiName"
android:layout_width="fill_parent"
           android:layout_height="wrap_content"
android:text="AP Name" />

       <TextView android:id="@+id/tvWifiMac"
android:layout_width="fill_parent"
           android:layout_height="wrap_content"
android:textSize="10dp"
           android:text="00:00:00:00:00:00" />

   </LinearLayout>

</LinearLayout>
```

Notre main.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   >
   <Button android:id="@+id/buttonRefresh"
android:text="Rechercher"
android:layout_width="fill_parent"
android:layout_height="wrap_content"></Button>
   <ListView android:id="@+id/listViewWifi"
android:layout_height="fill_parent"
android:layout_width="match_parent"></ListView>
</LinearLayout>
```

La classe représentant notre item pour la listview :

```
public class WifiItem {
   private String APName;
   private String AdresseMac;
   private int ForceSignal;

   public String getAPName() {
       return APName;
   }
   public void setAPName(String aPName) {
       APName = aPName;
   }
   public String getAdresseMac() {
       return AdresseMac;
   }
   public void setAdresseMac(String adresseMac) {
       AdresseMac = adresseMac;
   }
   public int getForceSignal() {
       return ForceSignal;
   }
   public void setForceSignal(int forceSignal) {
       ForceSignal = forceSignal;
   }

}
```

Notre adapter :

```
public class WifiAdapter extends BaseAdapter {

   private List<WifiItem> listeWifiItem;
   private LayoutInflater layoutInflater;

   public WifiAdapter(Context context,
List<WifiItem> objects) {

           listeWifiItem = objects;
           layoutInflater =
LayoutInflater.from(context);
   }

   public int getCount() {
       return listeWifiItem.size();
   }

   public Object getItem(int position) {
       return listeWifiItem.get(position);
   }
```

```
    public long getItemId(int position) {
        return position;
    }

    private class ViewWifiHolder {
        TextView tvApName;
        TextView tvAdresseMac;
        TextView ForceSignal;
    }

    public View getView(int position, View
convertView, ViewGroup parent) {
        ViewWifiHolder viewHolder;

        if(convertView == null) {
            viewHolder = new ViewWifiHolder();

            convertView =
layoutInflater.inflate(R.layout.item_wifi, null);

            viewHolder.tvApName = (TextView)
convertView.findViewById(R.id.tvWifiName);
            viewHolder.tvAdresseMac = (TextView)
convertView.findViewById(R.id.tvWifiMac);
            viewHolder.ForceSignal = (TextView)
convertView.findViewById(R.id.ForceSignal);

            convertView.setTag(viewHolder);
        } else {
            viewHolder =
(ViewWifiHolder)convertView.getTag();
        }

        // On affecte les valeurs

viewHolder.tvApName.setText(listeWifiItem.get(pos
ition).getAPName());

viewHolder.tvAdresseMac.setText(listeWifiItem.get
(position).getAdresseMac());

        // On change la couleur en fonction de la
force du signal

if(listeWifiItem.get(position).getForceSignal()
<= -80) {

viewHolder.ForceSignal.setBackgroundColor(Color.G
REEN);
        } else
if(listeWifiItem.get(position).getForceSignal()
<= -50) {

viewHolder.ForceSignal.setBackgroundColor(Color.Y
ELLOW);
        } else {

viewHolder.ForceSignal.setBackgroundColor(Color.R
ED);
        }

        return convertView;
    }

}
```

4. Implémentation

Passons désormais à l'implémentation de notre code, tout d'abord nous allons commencer par instancier tous les objets dont nous avons besoin.

```
private Button boutonRechercher;
private ListView listeViewWifi;
private List<WifiItem> listeWifiItem;
private WifiAdapter wifiAdapter;
private WifiManager wifiManager;
private WifiBroadcastReceiver broadcastReceiver;
```

- Le bouton de l'IHM.
- La listview de l'IHM.
- La liste des WifiItems.
- Notre adapter.
- Le WiFi Manager du sdk Android.
- Le broadcast receiver que nous allons créer par la suite.

Notre méthode onCreate :

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    listeViewWifi = (ListView)
findViewById(R.id.listViewWifi);
    boutonRechercher = (Button)
findViewById(R.id.buttonRefresh);

    boutonRechercher.setOnClickListener(new
OnClickListener() {

        public void onClick(View v) {
            if(wifiManager != null)
                wifiManager.startScan();
        }
    });

    // On récupère le service WiFi d'Android
    wifiManager = (WifiManager)
this.getSystemService(Context.WIFI_SERVICE);

    // Gestion de la liste des AP WiFi (voir tuto
sur les adapters et les
    // listviews)
    listeWifiItem = new ArrayList<WifiItem>();
    wifiAdapter = new WifiAdapter(this,
listeWifiItem);
    listeViewWifi.setAdapter(wifiAdapter);

    // Création du broadcast Receiver
    broadcastReceiver = new
WifiBroadcastReceiver();

    // On attache le receiver au scan result
    registerReceiver(broadcastReceiver, new
IntentFilter(

WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));

}
```

On commence par récupérer le bouton et la listview, ensuite on va lier notre WiFi Manager au service Android grâce au getSystemService.
Continuons en initialisant la liste d'objets et l' « adapter ».
Pour finir, on va instancier notre broadcast receiver et l'enregistrer pour qu'il soit appelé à chaque fois qu'un scan des réseaux Wi-Fi est fait.

```
// On arrête le receiver quand on met en pause
l'application
@Override
protected void onPause() {
   unregisterReceiver(broadcastReceiver);
   super.onPause();
}

// On remet en route le receiver quand on revient
sur l'application
@Override
protected void onResume() {
   registerReceiver(broadcastReceiver, new
IntentFilter(

WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
   super.onResume();
}
```

N'oublions pas ces deux parties qui permettent d'arrêter de recevoir les scans si vous avez mis l'application en arrière-plan et de nouveau recevoir les scans si vous revenez sur celle-ci.
Pour terminer, on fait des getters sur certaines de nos variables pour qu'elles soient accessibles dans le BroadcastReceiver.

```
public WifiManager getCurrentWifiManager() {
   return wifiManager;
}

public WifiAdapter getWifiAdapter() {
   return wifiAdapter;
}

public List<WifiItem> getListeWifiItem() {
   return listeWifiItem;
}
```

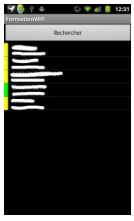
Passons maintenant à la création du BroadcastReceiver associé au Wi-Fi :

```
public class WifiBroadcastReceiver extends
BroadcastReceiver {

    private WifiManager wifiManager;
    private WifiAdapter wifiAdapter;
    private List<WifiItem> listeWifiItem;

    @Override
    public void onReceive(Context context, Intent
intent) {
        wifiManager = ((FormationWifiActivity)
context).getCurrentWifiManager();
        wifiAdapter = ((FormationWifiActivity)
context).getWifiAdapter();
        listeWifiItem = ((FormationWifiActivity)
context).getListeWifiItem();

        // On vérifie que notre objet est bien
instancié
        if (wifiManager != null) {

            // On vérifie que le WiFi est allumé
            if (wifiManager.isWifiEnabled()) {
                // On récupère les scans
                List<ScanResult> listeScan =
wifiManager.getScanResults();

                // On vide notre liste
                listeWifiItem.clear();

                // Pour chaque scan
                for (ScanResult scanResult :
listeScan) {
                    WifiItem item = new
WifiItem();


item.setAdresseMac(scanResult.BSSID);

item.setAPName(scanResult.SSID);

item.setForceSignal(scanResult.level);

                    Log.d("FormationWifi",
scanResult.SSID + " LEVEL "
                            + scanResult.level);

                    listeWifiItem.add(item);
                }

                // On rafraichit la liste

wifiAdapter.notifyDataSetChanged();
            } else {
                Toast.makeText(context, "Vous
devez activer votre WiFi",
                        Toast.LENGTH_SHORT);
            }
        }

    }

}
```

- on crée une classe qui hérite de BroadcastReceiver ;
- on redéfinit la méthode onReceive ;
- on récupère nos getters de l'activité ;
- on vérifie tout de même que le WiFi Manager est en route ;
- on vérifie aussi que le Wi-Fi est activé ;
- on récupère la liste des réseaux scannés ;
- on vide notre liste et on la remplit avec les nouvelles informations ;
- on spécifie à notre adapter que la liste a été modifiée.

5. Conclusion

Et voilà le tour est joué, normalement vous pouvez voir tous les réseaux Wi-Fi à portée de votre mobile.
Le broadcast receiver est appelé à chaque fois qu'un scan est fait par l'OS.



Retrouvez l'article d'Acesyde en ligne : Lien 11

Migration de données à partir du code Java : une approche dite code-first - Comment faciliter la migration de données lorsque l'on a généré son schéma à partir des entités métier

Cet article vous montrera comment faire évoluer votre schéma de base de données lorsque vous l'avez généré à partir des entités métier.

1. Introduction

Supposons que :

- vous avez développé une application en version V1 qui interagit avec une base de données relationnelle ;
- que vous avez livré cette application à votre client et que ce dernier l'a mise en production ;
- que quelques mois plus tard le client vous commande un gros lot d'évolutions (en somme une V2 de l'application) nécessitant de modifier le schéma de la base de données ;
- que vous avez généré le schéma de votre base de données à partir de vos entités métier (vous avez adopté une approche code-first).

Dans cette situation, vous aurez un script SQL à concevoir qui permette de faire passer le schéma de la base de données de production (qui est en version V1) en version V2. Tâche aussi délicate que fastidieuse, surtout lorsque le schéma contient beaucoup de tables.
Ce que je vous propose, c'est une méthode vous permettant de faciliter le passage du schéma V1 au schéma V2.
Dans cet article nous utiliserons :

- l'ORM Hibernate (Lien 12) ;
- la base de données relationnelle H2 (Lien 13) en version 1.3.167, particulièrement véloce en mode Embedded.

Cela dit, la démarche peut très bien s'appliquer à un projet utilisant un ORM comme EclipseLink, (ou OpenJPA, Entity Framework, etc.) et une base de données comme Oracle (ou MySQL, SQL Server, PostgreSQL, etc.).

2. Présentation du point de départ de la migration de données (V1)

Notre point de départ est une application bancaire avec deux classes : User et Account. Un utilisateur (User) peut avoir plusieurs comptes bancaires (Account), mais un compte bancaire ne peut être associé qu'à un seul utilisateur.

Classe User

```
package com.example.myproject.server.domain;

import java.io.Serializable;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Version;

@Entity
public class User implements Serializable{
    @Id
    @GeneratedValue
    private Long id;

     @Version
    private long version;
     @Column(length=12,nullable=false)
     private String fristName;
     @Column(length=12,nullable=false)
     private String lastName;
     @Column(nullable=false)
     private short age;


@OneToMany(mappedBy="user",cascade=CascadeType.ALL)
     private Set<Account> accounts;
}
```

Classe Account

```
package com.example.myproject.server.domain;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.Version;

@Entity
public class Account implements Serializable{
    @Id
    @GeneratedValue
    private Long id;
     @Version
    private long version;
     @Column(scale=2,nullable=false)
     private double amount;
     @ManyToOne(optional=false)
     private User user;
}
```

En utilisant les outils de génération de schéma d'Hibernate on obtient le script suivant :

Script de la base de données v1

```
    create table Account (
        id bigint generated by default as
identity (start with 1),
        amount double not null,
        version bigint not null,
        user_id bigint not null,
        primary key (id)
    );

    create table User (
        id bigint generated by default as
identity (start with 1),
        age smallint not null,
        fristName varchar(12) not null,
        lastName varchar(12) not null,
        version bigint not null,
        primary key (id)
    );

    alter table Account
        add constraint FK1D0C220D8CA544AB
        foreign key (user_id)
        references User;
```

Ce schéma a été livré au client, et la base de données de production repose le script SQL ci-dessus.

3. Présentation du point d'arrivée de la migration de données (V2)

Notre point d'arrivée se compose des mêmes classes que celles de la V1 :

Classe User

```
package com.example.myproject.server.domain;

import java.io.Serializable;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Version;

@Entity
public class User implements Serializable{
    @Id
    @GeneratedValue
    private Long id;

     @Version
    private long version;


//modified
     @Column(length=30,nullable=false)
     private String firstName;

//modified
     @Column(length=30,nullable=false)
     private String lastName;

     @Column(nullable=false)
     private short age;

//added
     @Column(length=10)
     private String phoneNumber;


@OneToMany(mappedBy="user",cascade=CascadeType.ALL)
     private Set<Account> accounts;
}
```

Classe Account

```
package com.example.myproject.server.domain;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.Version;

@Entity
public class Account implements Serializable{

    private static final long serialVersionUID =
8204815540254878355L;
    @Id
    @GeneratedValue
    private Long id;

     @Version
    private long version;

     @Column(scale=2,nullable=false)
     private double amount;

     @ManyToOne(optional=false)
     private User user;

//added
     @Column(nullable=false)
     private boolean isActive;
}
```

Vous constaterez que les classes ont été modifiées.

En utilisant les outils de génération de schéma d'Hibernate on obtient le script suivant :

Script de la base de données v2

```
    create table Account (
        id bigint generated by default as
identity (start with 1),
        amount double not null,
        isActive bit not null,
        version bigint not null,
        user_id bigint not null,
        primary key (id)
    );

    create table User (
        id bigint generated by default as
identity (start with 1),
        age smallint not null,
        firstName varchar(30) not null,
        lastName varchar(30) not null,
```

```
    phoneNumber varchar(10),
    version bigint not null,
    primary key (id)
);

alter table Account
    add constraint FK1D0C220D8CA544AB
    foreign key (user_id)
    references User;
```

On commence ici à cerner le problème de la migration de données. L'outil de génération d'Hibernate (tout comme celui d'EclipseLink) nous fournit un script permettant de générer le schéma V2 à partir de rien, mais pas de passer (de manière fiable) du schéma V1 au schéma V2.
Plutôt que d'effectuer un renommage, nous aurions pu nous contenter de rajouter de nouveaux champs et d'annoter les anciens en « Deprecated ». Ce procédé a au moins un avantage : on est sûr de ne perdre aucune donnée tant que l'on conserve le champ « Deprecated ». Un inconvénient est que l'on a du coup deux champs à gérer pour la lecture des données (le champ « Deprecated » pour les anciennes données, et l'autre pour les nouvelles données), et c'est dans l'hypothèse où l'on n'utilise que le nouveau champ pour l'écriture des données. Un autre inconvénient est que le fait de garder le champ « Deprecated » fait courir le risque que ce champ soit toujours utilisé. Un dernier inconvénient est que conserver un champ « Deprecated » ne fait que reculer l'échéance, puisque ces champs sont voués à être supprimés tôt ou tard.

4. Passage du point de départ au point d'arrivée

Pour passer du schéma V1 au schéma V2 nous aurons besoin d'un outil supplémentaire : Liquibase (Lien 14). Mais qu'est-ce que Liquibase ? C'est un outil open source, multiplateforme et multiSGBDR (système de gestion de bases de données relationnelles) qui permet de gérer les évolutions des schémas des bases de données. Voyons comment il peut nous aider à résoudre notre problème.

Tout d'abord téléchargez Liquibase, installez-le où bon vous semble et référencez son répertoire dans la variable d'environnement « Path » (%Path% sous Windows, $Path sous Linux).

Ce que nous allons faire dans un premier temps, c'est générer un « diff » entre les deux schémas, c'est-à-dire un script SQL permettant de passer du schéma V2 à partir du schéma V1.
En supposant que :

- votre driver JDBC d'H2 se trouve dans le répertoire D:/h2/bin/ ;
- que votre base (utilisée dans le développement du lot 1) V1 a pour URL JDBC jdbc:h2:tcp://localhost/file:D:/databases/liquibasev1 ;
- que votre base (utilisée dans le développement du lot 2) V2 a pour URL JDBC jdbc:h2:tcp://localhost/file:D:/databases/liquibasev2 .

Voici la commande à saisir dans votre shell :

```
> liquibase --defaultsFile=liquibase.properties
 diffChangeLog
```

Le fichier de propriétés « liquibase.properties » référencé dans la commande contient les informations permettant à Liquibase de se connecter à la base de données. Cela évite de les préciser à chaque ligne de commande.

Fichier liquibase.properties

```
#liquibase.properties
driver: org.h2.Driver
classpath: D:/h2/bin/h2-1.3.167.jar
url:
jdbc:h2:tcp://localhost/file:D:/databases/liquibasev1
username: sa
password:
referenceUsername: sa
referencePassword:
referenceUrl:
jdbc:h2:tcp://localhost/file:D:/databases/liquibasev2
```

La commande à proprement parler vous génère du XML, copiez la partie XML et mettez-le dans un fichier intitulé « changelog.xml ».

Fichier changelog.xml généré par Liquibase

```
<?xml version="1.0" encoding="UTF-8"
standalone="no"?>
<databaseChangeLog
xmlns="http://www.liquibase.org/xml/ns/dbchangelog" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocat
ion="http://www.liquibase.org/xml/ns/dbchangelog
http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-2.0.xsd">
    <changeSet author="RonnyGuillaume
(generated)" id="1354569877499-1">
        <addColumn tableName="ACCOUNT">
            <column name="ISACTIVE"
type="BOOLEAN">
                <constraints nullable="false"/>
            </column>
        </addColumn>
    </changeSet>
    <changeSet author="RonnyGuillaume
(generated)" id="1354569877499-2">
        <addColumn tableName="USER">
            <column name="FIRSTNAME"
type="VARCHAR(30)">
                <constraints nullable="false"/>
            </column>
        </addColumn>
    </changeSet>
    <changeSet author="RonnyGuillaume
(generated)" id="1354569877499-3">
        <addColumn tableName="USER">
            <column name="PHONENUMBER"
type="VARCHAR(10)"/>
        </addColumn>
    </changeSet>
    <changeSet author="RonnyGuillaume
(generated)" id="1354569877499-4">
        <modifyDataType columnName="LASTNAME"
newDataType="VARCHAR(30)" tableName="USER"/>
    </changeSet>
    <changeSet author="RonnyGuillaume
(generated)" id="1354569877499-5">
        <dropColumn columnName="FRISTNAME"
tableName="USER"/>
```

```
    </changeSet>
</databaseChangeLog>
```

Ce fichier XML contient l'ensemble des modifications qui ont « été faites » entre le schéma de la V1 et celui de la V2.

On constate un problème : la suppression de la colonne « fristname » de la table « User ». Vous avez dans votre version V1 considéré que la colonne « fristname » de la table « User » contiendrait le prénom des utilisateurs. Dans la version V2, vous vous êtes rendu compte de votre faute de frappe et vous l'avez renommée en « firstname ». Liquibase a constaté que la colonne « fristname » a disparu, il a donc supposé que vous l'avez supprimée. Liquibase a constaté que la colonne « firstname » est apparue, il a donc supposé que vous l'avez rajoutée. Toutes les données qui étaient présentes dans la colonne « fristname » sont donc supprimées. C'est un problème majeur des diff de schémas, les renommages ne sont perçus que comme des ajouts/suppressions de colonnes pas comme des renommages de colonnes. Dans ce cas-là il n'y a pas de miracle, il faut remplacer ces ajouts/suppressions « par des renommages à la main » dans le fichier XML. Il y a d'autres cas de figure qui ne sont pas supportés par Liquibase, je vous conseille d'aller lire la documentation officielle pour avoir plus d'informations à ce sujet.

Renommage dans le fichier changelog.xml :

Fichier changelog modifié

```
<?xml version="1.0" encoding="UTF-8"
standalone="no"?>
<databaseChangeLog
xmlns="http://www.liquibase.org/xml/ns/dbchangelo
g" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:schemaLocation="http://www.liquibase.org/xml/
ns/dbchangelog
http://www.liquibase.org/xml/ns/dbchangelog/dbcha
ngelog-2.0.xsd">
    <changeSet author="RonnyGuillaume
(generated)" id="1354569877499-1">
        <addColumn tableName="ACCOUNT">
            <column name="ISACTIVE"
type="BOOLEAN">
                <constraints nullable="false"/>
            </column>
        </addColumn>
    </changeSet>
    <changeSet author="RonnyGuillaume
(generated)" id="1354569877499-2">
<!-- modified -->
        <renameColumn tableName="USER"
oldColumnName="FRISTNAME"
newColumnName="FIRSTNAME"/>
    </changeSet>
    <changeSet author="RonnyGuillaume
(generated)" id="1354569877499-3">
        <addColumn tableName="USER">
            <column name="PHONENUMBER"
type="VARCHAR(10)"/>
        </addColumn>
    </changeSet>
    <changeSet author="RonnyGuillaume
(generated)" id="1354569877499-4">
        <modifyDataType columnName="LASTNAME"
newDataType="VARCHAR(30)" tableName="USER"/>
    </changeSet>
    <changeSet author="RonnyGuillaume
(generated)" id="1354569877499-5">
<!-- modified -->
        <modifyDataType columnName="FIRSTNAME"
newDataType="VARCHAR(30)" tableName="USER"/>
    </changeSet>
</databaseChangeLog>
```

Si vous n'êtes pas fan du XML, il est possible de faire ça en SQL. Il vous faut donc convertir le fichier XML en SQL :

```
>liquibase.bat
--defaultsFile=liquibase.properties
--changeLogFile=changelog.xml    updateSQL >
changelog.sql
```

Voici le fichier SQL généré :

Fichier changelog généré par Liquibase

```
--
*****************************************************
-- Update Database Script
--
*****************************************************
-- Change Log: changelog.xml
-- Ran at: 03/12/12 22:36
-- Against:
SA@jdbc:h2:tcp://localhost/file:D:/databases/liqu
ibasev1
-- Liquibase version: 2.0.5
--
*****************************************************

-- Create Database Lock Table
CREATE TABLE DATABASECHANGELOGLOCK (ID INT NOT
NULL, LOCKED BOOLEAN NOT NULL, LOCKGRANTED
TIMESTAMP, LOCKEDBY VARCHAR(255), CONSTRAINT
PK_DATABASECHANGELOGLOCK PRIMARY KEY (ID));

INSERT INTO DATABASECHANGELOGLOCK (ID, LOCKED)
VALUES (1, FALSE);

-- Lock Database
-- Create Database Change Log Table
CREATE TABLE DATABASECHANGELOG (ID VARCHAR(63)
NOT NULL, AUTHOR VARCHAR(63) NOT NULL, FILENAME
VARCHAR(200) NOT NULL, DATEEXECUTED TIMESTAMP NOT
NULL, ORDEREXECUTED INT NOT NULL, EXECTYPE
VARCHAR(10) NOT NULL, MD5SUM VARCHAR(35),
DESCRIPTION VARCHAR(255), COMMENTS VARCHAR(255),
TAG VARCHAR(255), LIQUIBASE VARCHAR(20),
CONSTRAINT PK_DATABASECHANGELOG PRIMARY KEY (ID,
AUTHOR, FILENAME));

-- Changeset changelog.xml::1354569877499-
1::RonnyGuillaume (generated)::(Checksum:
3:b2c07909d6104b5e05f53b7ca97dd2f2)
ALTER TABLE ACCOUNT ADD ISACTIVE BOOLEAN NOT
NULL;

INSERT INTO DATABASECHANGELOG (AUTHOR, COMMENTS,
DATEEXECUTED, DESCRIPTION, EXECTYPE, FILENAME,
ID, LIQUIBASE, MD5SUM, ORDEREXECUTED) VALUES
('RonnyGuillaume (generated)', '', NOW(), 'Add
Column', 'EXECUTED', 'changelog.xml',
'1354569877499-1', '2.0.5',
'3:b2c07909d6104b5e05f53b7ca97dd2f2', 1);

-- Changeset changelog.xml::1354569877499-
2::RonnyGuillaume (generated)::(Checksum:
3:8f11e99ddc30f9ac73330fed1cbb3dbb)
```

```
ALTER TABLE USER ALTER COLUMN FRISTNAME RENAME TO
FIRSTNAME;

INSERT INTO DATABASECHANGELOG (AUTHOR, COMMENTS,
DATEEXECUTED, DESCRIPTION, EXECTYPE, FILENAME,
ID, LIQUIBASE, MD5SUM, ORDEREXECUTED) VALUES
('RonnyGuillaume (generated)', '', NOW(), 'Rename
Column', 'EXECUTED', 'changelog.xml',
'1354569877499-2', '2.0.5',
'3:8f11e99ddc30f9ac73330fed1cbb3dbb', 2);

-- Changeset changelog.xml::1354569877499-
3::RonnyGuillaume (generated)::(Checksum:
3:673c4fdfd2fb3be7b6b194ea045eaf5c)
ALTER TABLE USER ADD PHONENUMBER VARCHAR(10);

INSERT INTO DATABASECHANGELOG (AUTHOR, COMMENTS,
DATEEXECUTED, DESCRIPTION, EXECTYPE, FILENAME,
ID, LIQUIBASE, MD5SUM, ORDEREXECUTED) VALUES
('RonnyGuillaume (generated)', '', NOW(), 'Add
Column', 'EXECUTED', 'changelog.xml',
'1354569877499-3', '2.0.5',
'3:673c4fdfd2fb3be7b6b194ea045eaf5c', 3);

-- Changeset changelog.xml::1354569877499-
4::RonnyGuillaume (generated)::(Checksum:
3:bdafbf8bdf6583983df82fe7c772881f)
ALTER TABLE USER ALTER COLUMN LASTNAME
VARCHAR(30);

INSERT INTO DATABASECHANGELOG (AUTHOR, COMMENTS,
DATEEXECUTED, DESCRIPTION, EXECTYPE, FILENAME,
ID, LIQUIBASE, MD5SUM, ORDEREXECUTED) VALUES
('RonnyGuillaume (generated)', '', NOW(), 'Modify
data type', 'EXECUTED', 'changelog.xml',
'1354569877499-4', '2.0.5',
'3:bdafbf8bdf6583983df82fe7c772881f', 4);

-- Changeset changelog.xml::1354569877499-
5::RonnyGuillaume (generated)::(Checksum:
3:5c740fc978ac3f362d111640113820cc)
ALTER TABLE USER ALTER COLUMN FIRSTNAME
VARCHAR(30);

INSERT INTO DATABASECHANGELOG (AUTHOR, COMMENTS,
DATEEXECUTED, DESCRIPTION, EXECTYPE, FILENAME,
ID, LIQUIBASE, MD5SUM, ORDEREXECUTED) VALUES
('RonnyGuillaume (generated)', '', NOW(), 'Modify
data type', 'EXECUTED', 'changelog.xml',
'1354569877499-5', '2.0.5',
'3:5c740fc978ac3f362d111640113820cc', 5);
```

Ce sont les commandes SQL qui auraient été exécutées par Liquibase si la ligne de commande était :

```
>liquibase.bat
--defaultsFile=liquibase.properties
--changeLogFile=changelog.xml   update
```

Vous constaterez qu'il y a des commandes SQL parasites concernant une certaine table « DATABASECHANGELOG », si vous vous contentez de faire du diff entre deux schémas, supprimez ces commandes, dans le cas contraire je vous conseille de lire la rubrique suivante.

Le fichier « changelog.sql » contient désormais toutes les commandes SQL pour passer d'un schéma V1 à un schéma V2.
Comme vous avez pu le constater, la génération d'un diff est simple mais nécessite quelques modifications manuelles :

- au minimum la suppression de tout ce qui concerne les changelog de Liquibase (DATABASECHANGELOG, etc.) ;
- dans le pire des cas, correction des renommages de colonnes et de tout ce qui n'est pas géré intelligemment par Liquibase.

5. Pour aller plus loin : gestion des schémas

Certes, Liquibase ne se contente pas de faire des diff. Il permet de gérer les versions de votre schéma. Hélas, il ne permet pas de migrer votre schéma d'une V1 à une V3 (version 3 découlant d'un lot 3) correctement : il le fait sans passer par la V2…
C'est pour cela que je vous conseille d'utiliser MyBatis Migrations (Lien 15) pour la gestion des versions de schéma.
Sachez que l'utilisation de MyBatis Migrations n'oblige pas à utiliser le framework de persistance MyBatis. Je ne vais pas vous expliquer en détail comment utiliser MyBatis Migrations, la documentation officielle le fait très bien, je vais juste vous expliquer comment utiliser les scripts de diff générés par Liquibase avec MyBatis Migrations.

Installez MyBatis Migrations conformément à la documentation officielle :

- téléchargez l'archive de MyBatis Migrations et décompressez-la où bon vous semble ;
- créez une variable d'environnement « MIGRATIONS_HOME » qui a pour valeur l'emplacement du répertoire d'installation de MyBatis Migrations (exemple : « D:\migration_home ») ;
- rajoutez dans la variable Path, le répertoire « bin » du répertoire d'installation de MyBatis Migrations ;
- lancez un shell et créez votre « repository » en tapant la commande suivante :

```
> migrate --path=repertoire init
```

Où « repertoire » est un répertoire pris en compte par votre gestionnaire de versions (Git, SVN, CVS ou autre) .Ceci permettra à toute votre équipe de développement de partager le même repository ;

- mettez le driver JDBC (dans notre cas h2-1.3.167.jar) dans le répertoire « driver » du repository nouvellement créé ;
- allez dans le répertoire « environments » de votre repository, ouvrez le fichier « development.properties » et saisissez les informations permettant à MyBatis Migrations de se connecter à la base de données :

Fichier development.properties

```
## Base time zone to ensure times are consistent
across machines
time_zone=GMT+0:00

## The character set that scripts are encoded
with
# script_char_set=UTF-8

## JDBC connection properties.
driver=org.h2.Driver
```

```
url=jdbc:h2:tcp://localhost/file:D:/databases/liquibasev1
username=sa
password=

#
# A NOTE ON STORED PROCEDURES AND DELIMITERS
#
# Stored procedures and functions commonly have nested delimiters
# that conflict with the schema migration parsing.  If you tend
# to use procs, functions, triggers or anything that could create
# this situation, then you may want to experiment with
# send_full_script=true (preferred), or if you can't use
# send_full_script, then you may have to resort to a full
# line delimiter such as "GO" or "/" or "!RUN!".
#
# Also play with the autocommit settings, as some drivers
# or databases don't support creating procs, functions or
# even tables in a transaction, and others require it.
#

# This ignores the line delimiters and
# simply sends the entire script at once.
# Use with JDBC drivers that can accept large
# blocks of delimited text at once.
send_full_script=true

# This controls how statements are delimited.
# By default statements are delimited by an
# end of line semicolon.  Some databases may
# (e.g. MS SQL Server) may require a full line
# delimiter such as GO.
# These are ignored if send_full_script is true.
delimiter=;
full_line_delimiter=false

# If set to true, each statement is isolated
# in its own transaction.  Otherwise the entire
# script is executed in one transaction.
# Few databases should need this set to true,
# but some do.
auto_commit=false

# Custom driver path to allow you to centralize your driver files
# Default requires the drivers to be in the drivers directory of your
# initialized migration directory (created with "migrate init")
# driver_path=

# Name of the table that tracks changes to the database
changelog=CHANGELOG

# Migrations support variable substitutions in the form of ${variable}
# in the migration scripts.  All of the above properties will be ignored though,
# with the exception of changelog.
# Example: The following would be referenced in a migration file as ${ip_address}
# ip_address=192.168.0.1
```

```
> migrate new « v1 »
```

Ceci aura pour effet de créer un fichier SQL censé contenir tout ce qui est nécessaire pour créer le schéma V1.
Ce que vous aurez à mettre dans le script SQL qui porte un nom comme « 20121015120019_v1.sql » (présent dans le répertoire « scripts » de votre repository) est le schéma de création de la base de données du lot 1.

Lorsque vous livrerez votre version V2 vous aurez à saisir dans votre shell la commande suivante :

```
> migrate new « v2 »
```

Ceci aura pour effet de créer un fichier SQL censé contenir tout ce qui est nécessaire pour passer du schéma V1 au schéma V2.
Ce que vous aurez à mettre dans le script SQL qui porte un nom comme « 20121015122822_v2.sql » (présent dans le répertoire « scripts » de votre repository) est donc le script « changelog.sql ».

Étant donné que Liquibase et MyBatis Migrations sont deux outils de gestion de versions, pourquoi les utiliser conjointement ? Parce que d'une part comme je l'ai dit plus tôt les versions sont mal gérées avec Liquibase et d'autre part MyBatis Migrations ne gère pas les diff.

Je vous conseille donc d'appliquer le processus de gestion de schéma suivant dans vos projets :

1. Générez vos diff avec Liquibase ;
2. Corrigez le script généré en supprimant toute notion de versionnage créée par Liquibase (création de table DATABASECHANGELOG, etc.), réglez les problèmes de renommage, etc. ;
3. Créez une nouvelle version de votre schéma avec MyBatis Migrations via la commande ‘ migrate new « nom_de_la_version » ‘

Ainsi, si vous disposez d'un schéma (quelle que soit sa version) et d'un repository, la saisie dans un shell de la commande suivante :

```
> migrate up
```

vous permettra de le mettre à jour correctement. C'est-à-dire que seuls les scripts de diff (générés par Liquibase et corrigés par vous) nécessaires seront passés. Vous n'avez donc plus à vous préoccuper de la version de tel ou tel schéma en production pour savoir comment le mettre à jour.

6. Conclusion

Avec cette méthode, les migrations de données prendront moins de temps, car une bonne partie du diff sera fait par Liquibase. Cela réduit d'autant le nombre d'erreurs possibles puisqu'une bonne partie du travail est désormais automatisée.

Retrouvez l'article de Ronny Guillaume en ligne : Lien 16

L'injection dans Eclipse 4

Le but de cet article est de démystifier le mécanisme d'injection et de décrire les annotations associées et utilisées dans Eclipse 4.

Le comportement implicite de ces annotations est également décrit.

1. Introduction

Dans tous les tutoriels que l'on peut trouver sur Eclipse 4, beaucoup considèrent que le lecteur manipule depuis longtemps les annotations et l'injection, et peu d'informations sont données sur le fonctionnement interne de tous ces mécanismes.
Le but de ce premier article est de démystifier les annotations et leur fonctionnement, et notamment celles qui concernent le mécanisme de base de l'injection.
Je détaillerai ensuite dans un autre article comment utiliser les autres annotations.
J'espère qu'il vous aidera à mieux comprendre la puissance de ce concept et l'intérêt de les utiliser dans Eclipse 4.

2. Qu'est-ce qu'une annotation ?

L'annotation est définie par le langage Java dès sa version 1.5. Il s'agit simplement d'un élément du langage qui peut être associé à une classe, une méthode, un constructeur, un champ ou un paramètre.
Elle se définit dans un fichier Java du nom de l'annotation et se déclare avec la notation @interface. L'annotation doit aussi définir sa cible d'application directement dans sa définition. Par exemple l'annotation d'injection est définie de la manière suivante :

```
@Target({ METHOD, CONSTRUCTOR, FIELD })
@Retention(RUNTIME)
@Documented
public @interface Inject {}
```

Cette définition se fait également par des annotations réservées pour les annotations !
Il est également possible de définir des annotations avec des paramètres, en créant à l'intérieur des méthodes qui donnent le type de la valeur et leur valeur par défaut. Par exemple, l'annotation de préférence d'Eclipse 4 est définie de la manière suivante :

```
@Qualifier
@Documented
@Target({ElementType.FIELD,
ElementType.PARAMETER})
@Retention(RetentionPolicy.RUNTIME)
public @interface Preference {
    String value() default ""; // key in the node

    String nodePath() default "";
}
```

Je ne développerai pas plus la déclaration des annotations en Java, de nombreux articles le font déjà, l'essentiel étant de retenir qu'une annotation s'applique à une cible et peut donc être utilisée par la suite, en utilisant l'introspection Java.

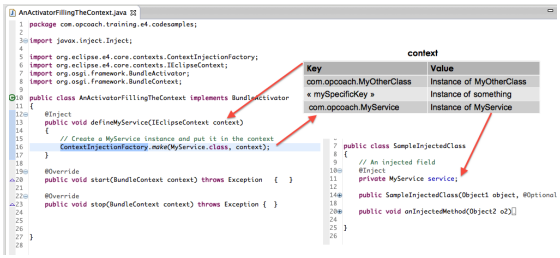
3. Le rôle de l'introspection Java.

Les annotations n'ont aucun comportement dynamique associé dans le langage. Pour leur donner un sens, il faut qu'à un moment donné, la classe contenant les annotations soit introspectée pour analyser ses annotations et pour lui appliquer un traitement particulier.
Ce traitement peut, par exemple, intervenir lors de la compilation de la classe pour générer du code complémentaire. Dans ce cas, c'est le compilateur qui analyse le code et effectue le comportement.
Il faut donc bien comprendre que **les annotations ne sont rien sans un traitement derrière qu'il faut effectuer**.
C'est donc, lors du runtime, qu'Eclipse 4 va exploiter les classes annotées qu'il va traiter au moment voulu !
Toute la difficulté est donc de connaître ces moments et de savoir à quoi vous avez accès. Eclipse 4 dispose d'un **renderer** qui permet d'afficher votre application décrite dans le modèle d'application.
C'est donc ce renderer qui va analyser la classe traitée à un moment donné (par exemple un Part est analysé au moment de l'affichage d'une perspective), et qui va ensuite appeler les méthodes possédant telle ou telle annotation.
Tout se passe donc par introspection et la logique séquentielle de votre programme va en être un peu bouleversée (la recherche dans la pile d'appel lors du débogage aussi !).
Prenons l'exemple de l'annotation @PostConstruct, que l'on peut poser sur une méthode. Il faut savoir que cette méthode ne sera appelée qu'une fois et après que tous les comportements d'injection ont été appliqués (j'y reviendrai tout à l'heure).
Donc si on ne connaît pas la logique du moteur qui exploite les annotations, il va être difficile de comprendre le séquencement d'appel et les endroits où annoter.
Détaillons donc maintenant un peu plus le fonctionnement de l'annotation d'injection, qui est la base de tout le système, et qui est également implicitement utilisée par toutes les autres annotations ensuite.

4. L'annotation @Inject

Le mécanisme d'injection consiste à déléguer l'instanciation des objets à un injecteur qui aura pour rôle ensuite d'injecter les instances créées aux endroits voulus.
Comme je l'ai dit précédemment, cette annotation ne peut fonctionner que si elle est exploitée dans un contexte de

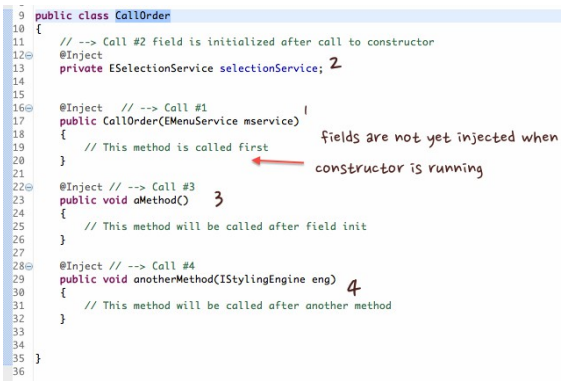
création par un injecteur.
Donc si vous avez écrit une classe avec des annotations @Inject, n'espérez pas que le mécanisme soit pris en charge, si vous instanciez votre classe avec un « new » traditionnel en java !
Il faudra appeler l'injecteur (Cf **ContextInjectionFactory**) pour construire chaque objet afin que ses annotations soient traitées. Le contexte d'injection peut alors être reçu par injection ! Le schéma suivant résume la situation :



Concernant la logique d'appel, l'annotation @Inject fonctionne de la manière suivante :

- Appel du constructeur,
- Initialisation des champs,
- Appel des méthodes injectées.

On obtient donc la situation :



Il faut faire attention, car la logique est encore subtile pour chacune des parties :

4.1. Appel du constructeur

Il se fait directement par l'injecteur. Mais quel constructeur choisir si plusieurs ont été définis puisqu'on ne l'appelle pas explicitement ?
Pour le constructeur, l'injecteur va choisir celui qui possède une annotation d'injection et qui possède le maximum de paramètres injectables. Ainsi si vous avez un constructeur avec 3 paramètres, dont 2 seulement peuvent être injectés, ce constructeur ne sera pas appelé.
Autre petite caractéristique qui a son importance, **le constructeur n'a jamais accès aux champs injectés de la classe,** car ils seront initialisés après son appel. Par contre, il peut manipuler les champs qui ne possèdent pas d'annotation @Inject.

4.2. Injection des champs

Chaque champ de la classe, précédé d'une annotation @Inject sera initialisé, une fois le constructeur appelé et ayant marché (s'il y a eu une exception, tout s'arrête).
Si un champ ne peut pas être injecté (valeur nulle), l'injecteur génère une exception. Si ce cas peut se produire, il faut précéder le champ de l'annotation @Optional.
Si votre objet a des champs hérités injectés, ces derniers seront initialisés avant.

4.3. Appel des méthodes injectées

L'instanciation de la classe possédant les annotations va se terminer par l'appel de chaque méthode possédant une annotation @Inject.
Tous les paramètres attendus par ces méthodes doivent pouvoir être injectés (à moins qu'ils soient précédés d'une annotation @Optional) sous peine d'exception.
L'ordre d'appel des méthodes injectées est indéterminé. Toutefois, les méthodes injectées héritées sont appelées avant celles de la classe.
Et voilà votre objet a été initialisé automatiquement et intégralement… mais ce n'est pas tout !
Un comportement supplémentaire à connaître :
C'est aussi là que l'injection apporte sa magie, car chaque méthode sera rappelée automatiquement, si l'un de ses paramètres change et ce à tout moment. De même un attribut injecté sera remis à jour automatiquement si la dernière valeur injectée a changé.
Ainsi si vous définissez une méthode @Inject dont l'un des paramètres est par exemple la sélection courante, cette méthode sera appelée à chaque modification de la sélection courante.
Ce mécanisme fonctionne avec tous les objets reçus et remplace partiellement le mécanisme des listeners, plus traditionnel en Eclipse 3.
Ainsi dans l'exemple suivant :

```
@Inject
public void setSelection(@Optional
@Named(IServiceConstants.ACTIVE_SELECTION) Object
o, Adapter adapter)
{
    Rental r = adapter.adapt(o, Rental.class);
    setRental(r);
}
```

La méthode sera appelée lors de la construction de l'objet qui la contient, mais également à chaque moment où un des deux objets passés en paramètre change de valeur dans l'injecteur. Pour ce cas, l'Adapter ne changera pas, car c'est un singleton dans Eclipse 4, par contre, la sélection active (passée avec une annotation @Named) changera régulièrement et sera automatiquement réinjectée dans cette méthode !
Remarque : On notera dans cet exemple que le paramètre injecté étant de type « Object », il doit être récupéré par un nom unique, qui, dans notre cas de sélection, est fourni par l'interface **IServiceConstants** d'Eclipse 4. Il serait aussi possible de remplacer le type Object attendu, par le type Rental. Dans ce cas, la méthode ne sera appelée que si la sélection courante est une instance de Rental.
Voilà donc l'une des subtilités de l'injection qui est aussi utilisée dans Eclipse 4 ! Et attention, là nous n'avons décrit que le mécanisme d'injection, nous n'avons pas encore parlé des annotations spécifiques utilisées par le renderer (objet d'un autre article).

5. Quatre annotations complémentaires pour finir.

Mais l'annotation @Inject seule ne suffit pas à couvrir tous

les cas. En effet, l'ordre d'appel des méthodes annotées avec @Inject n'étant pas déterminé, il peut être difficile de s'y retrouver. Certains paramètres également peuvent correspondre à plusieurs instances possibles dans l'injecteur, ou peuvent être optionnels. Le développeur dispose donc d'annotations complémentaires permettant de traiter ces différents cas.

5.1. @PostConstruct

Cette annotation ne s'applique que sur une méthode qui sera appelée une fois toutes les injections terminées. Cette annotation permet ainsi de finaliser l'initialisation d'un objet. Elle est utilisée par exemple pour créer le contenu d'une vue dans Eclipse 4, pour initialiser un Addon, pour brancher des listeners, etc.

5.2. @PreDestroy

Comme @PostConstruct, elle ne s'applique que sur une méthode qui sera appelée juste avant la destruction d'un objet. On peut se servir de cette annotation pour indiquer une méthode qui enlèvera les éventuels listeners ou qui gérera les subtilités de désallocation.

5.3. @Optional

À mettre devant un paramètre de méthode ou un champ de classe pour indiquer que l'on gère le fait qu'il n'y ait pas de valeur. N'oubliez pas que si aucune valeur ne peut être injectée, une exception sera levée.

5.4. @Named

Permet, associé à un nom passé en paramètre de l'annotation, de retrouver un objet nommé stocké dans l'injecteur. On utilise cette annotation quand le type recherché peut avoir plusieurs instances possibles.

5.5. Que peut-on injecter ?

Là, il faut aussi être intuitif et on peut arriver à s'en sortir presque à chaque fois avec un peu de réflexion (c'est le cas de le dire !).
Déjà il faut bien comprendre qu'Eclipse va gérer pour vous des contextes d'injection (**IEclipseContext**) hiérarchiques initialisés au fur et à mesure des traitements.
Par exemple, quand l'application démarre, un contexte global est créé. Ce contexte va contenir tous les services OSGi utilisés par Eclipse 4. Puis quand le renderer va s'exécuter, il va créer un contexte pour l'application model (fils du contexte principal). Ensuite, une perspective se crée, elle aura aussi son propre contexte relié au contexte parent. Idem pour le Part.
Donc, logiquement, le renderer va vous fournir les instances nécessaires pour que votre méthode puisse fonctionner.
C'est pour cela que si on injecte une instance de Component dans la méthode de création du contenu d'une vue, ce Component est forcément le parent dans lequel la vue doit se créer ! Ce n'est pas magique. C'est le comportement dynamique du renderer qui à un moment donné l'initialise pour vous.
Si vous arrivez ensuite dans un autre part, le Component injecté sera bien sur différent !
On peut donc se dire que si on a besoin de quelque chose à un endroit, normalement il est disponible !
Plus précisément, dans Eclipse 4, les objets fournissant un nouveau contexte dans le cadre de l'application model héritent de l'interface **MContext**. On trouve par exemple : **MApplication**, **MWindow**, **MPerspective**, **MPart**, **MPopupMenu**. Sur chacun de ces objets, on pourra récupérer le contexte en appelant la méthode **getContext**().
Dans un prochain article, je vous détaillerai les autres annotations d'Eclipse 4, et surtout le moment où celles-ci sont utilisées.

6. Conclusion

Comme on peut le constater, l'injection est un mécanisme très puissant, avec des comportements implicites qu'il faut bien maitriser. Une fois cette connaissance acquise, on découvre une nouvelle manière de programmer, plus souple et plus puissante, qui est la base du développement d'application Eclipse RCP version 4.

Retrouvez l'article d'Olivier Prouvost en ligne : Lien 17

NetBeans 7.3 sort en Release Candidate et intègre Easel, une plateforme pour le développement HTML5 et JavaScript dans un environnement Java

NetBeans 7.3, la prochaine mise à jour majeure de l'environnement de développement open source pour Java, C, C++, PHP, HTML, JavaScript, etc. est disponible en Release Candidate (RC).

Cette mouture accorde une place d'honneur à l'amélioration de la prise en charge des derniers standards du Web dont HTML5, CSS3 et JavaScript.

La nouveauté phare de Netbeans 7.3 est l'intégration du projet Easel, une plateforme qui permet le développement HTML5 et JavaScript dans un environnement Java. Easel dispose d'une interface d'édition pour HTML5, JavaScript et jQuery, avec le support de l'autocomplétion.

Easel dispose d'un nouveau débogueur JavaScript basé sur des API de débogage à distance de WebKit, mais il peut également s'intégrer avec le navigateur Chrome.

Le but du projet Easel est de permettre aux développeurs de créer rapidement des clients JavaScript (en utilisant MVC) pour consommer ou tester des services RESTful. Un nouveau modèle de projet optimisé pour les applications Web Easel est disponible avec NetBeans 7.3 RC.

À cette nouveauté majeure, s'ajoute un nouveau débogueur JavaScript basé sur le projet Nashorn (moteur d'exécution JavaScript développé en Java), une nouvelle barre de navigation, l'intégration d'un outil de test Java Persistence (JPQL) et des améliorations du processus de développement des services REST.

Il faut noter également le support du SDK JavaFX 2.2.4, y compris des améliorations de l'éditeur FXML/Scene Builder FXML, dont une meilleure précision de la complétion de code.

En fournissant un environnement unique où les développeurs peuvent créer des services Java et des clients HTML5, NetBeans 7.3 dispose des outils pour développer et déboguer des applications web et mobiles qui consomment les services Java en utilisant les derniers standards du web.

Télécharger NetBeans 7.3 RC : Lien 18

Commentez la news de Hinault Romaric en ligne : Lien 19.

Le mécanisme de sessions Rails

Cet article est publié avec l'aimable autorisation de Synbioz.

L'article original peut être lu sur le blog de Synbioz : Le mécanisme de sessions en rails Lien 20.

1. Le fonctionnement du HTTP

HTTP est un protocole sans état. Il n'inclut pas de notion d'identité de base et chaque requête est donc indépendante. Cela signifie que le serveur ne me reconnaît pas entre chaque requête.
C'est la base du fonctionnement du Web, et l'opposé d'une connexion client - serveur tel que FTP.
L'avantage est que le serveur n'a pas à gérer ses clients, tout le monde est logé à la même enseigne. L'inconvénient est que l'ajout d'un mécanisme d'identification va ajouter un surplus d'information à chaque requête.

2. Les sessions, ou comment transformer HTTP en protocole à état

Le mécanisme de sessions prend place à la fois côté client et serveur. Il est activé de base sur une application Rails par le biais d'un middleware (Lien 21) (ActionDispatch::Session::CookieStore) qui peut être désactivé au besoin.
Lorsqu'un nouvel utilisateur arrive sur l'application, si le serveur ne reçoit pas d'identifiant de session il va en créer un qu'il va transmettre dans sa réponse.
Côté client cet identifiant va être stocké dans un cookie. Les cookies ne sont pas forcément des fichiers plats. Sous Chrome les cookies sont stockés dans une base SQLite et sont limités à 4 ko en contenu.
Il est fortement déconseillé de placer des objets en session, car cela pourrait poser des soucis d'encodage/décodage.
Par exemple on mettra un id utilisateur plutôt que l'objet utilisateur. Qui plus est, cela permet de limiter l'espace occupé par le cookie.

3. Les sessions avec Ruby on Rails

Ruby on Rails possède plusieurs mécanismes de stockage de session. Par défaut il utilise le cookie store qui stocke tout sur le client et ne nécessite aucune configuration.
Dans ce cas l'identifiant de session n'a pas d'intérêt pour la partie serveur.
Ce paramètre peut être modifié dans config/initializers/session_store.rb.

```
App::Application.config.session_store
:cookie_store, :key => '_app_application_session'
```

Les autres mécanismes de stockage sont la base de données et le cache. Dans ces cas l'identifiant de session va permettre au serveur de retrouver les données associées en base ou en cache.
L'avantage de ces mécanismes est de pouvoir stocker plus de données voire d'améliorer les performances dans le cas du cache.
Attention toutefois, si votre serveur de cache vient à tomber vous n'avez plus d'utilisateurs connectés. D'autre part ces mécanismes impliquent une purge côté applicatif si vous ne voulez pas stocker des milliers de sessions fantômes.
Dernier point, vous aurez besoin de partager ces sessions entre serveurs si vous mettez en place du load balancing sur votre application.

4. Le contenu des cookies

Le contenu des cookies est encodé avec le secret de l'application Rails, qui peut être surchargé dans votre initializer avec l'option secret_token.
Voici un exemple de contenu de cookie :

```
BAh7CEkiD3Nlc3Npb25faWQGOgZFRkkiJWNhZTFlMjY0YWRlZ
Tc2NjNhZDc4YzY4YzkwMzk3NWVlBjsAVEkiEF9jc3JmX3Rva2
VuBjsARkkiMXBEZlQrdkVXZndjdGpQd1JSNTRQUjhYWlp6WHF
ldUlZYURZZklkYmh4UVE9BjsARkkiGXdhcmRlbi51c2VyLnVz
ZXIua2V5BjsAVFsISSIJVXNlcgY7AEZbBmkE8NcBLUkiIiQyY
SQxMCRMUXNTdU9rbk4wSVIySUZORHFESVcuBjsAVA==--
f58cf55b4d11b47c398103643b0ffe7ffdbff309
```

Le cookie présente deux parties, séparées par -, d'un côté le contenu et de l'autre côté la signature. Séparons-les :

```
  content =
"BAh7CEkiD3Nlc3Npb25faWQGOgZFRkkiJWNhZTFlMjY0YWRl
ZTc2NjNhZDc4YzY4YzkwMzk3NWVlBjsAVEkiEF9jc3JmX3Rva
2VuBjsARkkiMXBEZlQrdkVXZndjdGpQd1JSNTRQUjhYWlp6WH
FldUlZYURZZklkYmh4UVE9BjsARkkiGXdhcmRlbi51c2VyLnV
zZXIua2V5BjsAVFsISSIJVXNlcgY7AEZbBmkE8NcBLUkiIiQy
YSQxMCRMUXNTdU9rbk4wSVIySUZORHFESVcuBjsAVA=="
  signature =
"f58cf55b4d11b47c398103643b0ffe7ffdbff309"
```

À la réception du cookie, Rails va vérifier que le contenu et la signature correspondent. Pour cela il va utiliser son secret token.

Vous ne devez jamais divulguer ce token où vos sessions seront usurpables.

Voyons quel est le secret de notre application de démo :

```
secret = Rails.application.config.secret_token
"0b22c3a50d0ba81e396b880268d8d13c5980896a3761a24b
c87e704f3589f9d0ef2528ef7480101e0edeccc5320f6310d
54fd06ad7df574ee6e8e349ba01ace2"
```

Voilà ce que fait Rails pour générer la signature :

```
OpenSSL::HMAC.hexdigest(OpenSSL::Digest::SHA1.new
, secret, str)
"f58cf55b4d11b47c398103643b0ffe7ffdbff309"
```

Effectivement, on retombe bien sur notre signature.
Par le biais de ce mécanisme, modifier le contenu du cookie ne permet pas d'usurper une session, car la signature ne sera plus valable et notre *framework* s'en rendra compte.
Attention, si les cookies sont signés leur contenu n'est toutefois pas encrypté.
Il est tout à fait possible d'extraire le contenu d'un cookie.

```
session =
Marshal.load(Base64.decode64(CGI.unescape(content
)))
```

```
{
              "session_id" =>
"cae1e264adee7663ad78c68c903975ee",
             "_csrf_token" =>
"pDfT+vEWfwctjPwRR54PR8XZZzXqeuIYaDYfIdbhxQQ=",
    "warden.user.user.key" => [
        [0] "User",
        [1] [
            [0] 755095536
        ],
        [2] "$2a$10$LQsSuOknN0IR2IFNDqDIW."
    ]
}
```

J'espère que cet article vous aura permis de mieux appréhender le fonctionnement des sessions, notamment dans Ruby on Rails.

Retrouvez l'article de Synbioz en ligne : Lien 22

Gestion de sous-domaines avec Rails

Lors du développement d'un site ou d'une application web il est possible que vous ayez à gérer des sous-domaines. En effet, ces derniers peuvent servir à séparer différentes parties de votre site pour plus de clarté mais aussi pour des besoins de référencement par exemple.

Lors de mes développements en local j'utilise Pow (Lien 23), ce qui me permet d'avoir des URL du type mon-projet.dev et je peux donc avoir des sous-domaines sans avoir à modifier mon fichier /etc/hosts.

Cet article est publié avec l'aimable autorisation de Synbioz.

L'article original peut être lu sur le blog de Synbioz : Gestion de sous-domaines avec Rails (Lien 24).

1. Gestion des routes et des liens

Dans l'exemple choisi, nous allons développer une application qui gère des championnats sportifs et chaque équipe aura son sous-domaine.
Dans votre application Rails, après avoir généré les modèles et contrôleurs nécessaires (pensez à ajouter un champ subdomain dans votre modèle Team), il vous faut ajouter une constraint afin de gérer les sous-domaines.

```
MonApplication::Application.routes.draw do
  constraints(Subdomain) do
    match '/' => 'teams#show'
  end
end
```

Ensuite, il est nécessaire de récupérer le sous-domaine afin de savoir s'il faut le prendre en compte ou non. Pour cela, une classe Subdomain doit être ajoutée dans le répertoire lib/. Cette dernière renvoie un booléen afin de savoir si oui ou non le code contenu dans la constraint est exécuté.

```
class Subdomain
  def self.matches?(request)
    request.subdomain.present? &&
request.subdomain != 'www'
  end
end
```

Il est possible d'ajouter d'autres exceptions que « www » si certains sous-domaines sont réservés à d'autres usages par exemple.

En ce qui concerne les liens dans vos vues, il est nécessaire de passer en paramètre le sous-domaine lorsque cela est nécessaire. Pour cela, un helper peut être créé :

```
module UrlHelper
  def with_subdomain(subdomain)
    subdomain ||= ""
    subdomain += "." unless subdomain.empty?
    [subdomain, request.domain,
request.port_string].join
  end
end
```

Il suffit ensuite d'inclure ce helper dans votre ApplicationController pour qu'il soit disponible dans tous les contrôleurs du projet. Il est aussi possible de l'inclure uniquement dans certains contrôleurs si vous jugez cela plus pertinent.
Cela permet dans les vues de générer des liens de la façon suivante :

```
<%= link_to team.name, root_url(:host =>
with_subdomain(team.subdomain)) %>
```

Ce lien redirigera l'utilisateur vers la page show de l'équipe en question avec le sous-domaine correspondant.
Afin de faciliter l'usage des routes pour générer des liens, il est possible de surcharger la méthode url_for :

```
def url_for(options = nil)
  if options.kind_of?(Hash) && options.has_key?
```

```
(:subdomain)
    options[:host] =
with_subdomain(options.delete(:subdomain))
  end
  super
end
```

Grâce à cette méthode, il sera possible de passer un paramètre subdomain aux URL :

```
<%= link_to team.name, root_url(:subdomain =>
team.subdomain) %>
```

Afin de générer des URL sans aucun sous-domaine, il suffit de le préciser :

```
<%= link_to "Accueil", root_url(:subdomain =>
false) %>
```

Il se peut également qu'il y ait plusieurs niveaux de sous-domaines dans votre application. Afin de gérer cela, il suffit de modifier les méthodes créées précédemment pour spécifier quel niveau vous souhaitez (dans ce cas nous avons deux niveaux de sous-domaines) :

```
# /app/helpers/url_helper.rb
def with_subdomain(subdomain)
  subdomain ||= ""
  subdomain += "." unless subdomain.empty?
  [subdomain, request.domain(2),
request.port_string].join
end

# /lib/subdomain.rb
class Subdomain
  def self.matches?(request)
    request.subdomain(2).present? &&
request.subdomain(2) != "www"
  end
end
```

C'est donc le premier sous-domaine qui sera modifié dans ce cas et le second restera identique pour obtenir des URL du type one.test.myapp.dev ou two.test.myapp.dev.

2. Finder et sous-domaine

Une fois que vous avez réussi à créer vos routes et vos URL en gérant les sous-domaines, il faut les récupérer dans les contrôleurs. Dans notre cas, nous voulons trouver l'équipe (Team) correspondant au sous-domaine :

```
class TeamsController < ApplicationController
  def show
    @team =
Team.find_by_subdomain(request.subdomain)
  end
end
```

Les objets sont donc récupérés grâce à leur sous-domaine et non plus grâce à leur id et il vous est possible de créer un sous-domaine par équipe sur votre site et de récupérer l'équipe correspondante à chaque fois.
Vous pouvez ensuite gérer vos routes comme s'il s'agissait de simple nested resources :

```
constraints(Subdomain) do
  match '/' => 'groups#show'

  resources :players
  resources :games
end
```

En ajoutant un before_filter qui récupérera l'équipe grâce au sous-domaine, il n'y a aucune différence avec l'usage d'un id. Il faut tout de même bien vérifier que le champ subdomain est bien unique.

3. Validations

Pour notre modèle Team nous avons créé un champ subdomain afin de pouvoir utiliser un finder sur ce champ. Il peut s'avérer utile d'avoir des validations sur champ et on peut en trouver une très intéressante sur ce site : Lien 25.
Dans le modèle, il suffit d'ajouter la validation suivante :

```
validates :subdomain, presence: true, uniqueness:
true, subdomain: true
```

Il y a donc un validateur de sous-domaine. Il faut maintenant créer ce dernier. Pour cela, ajoutez un fichier subdomain_validator.rb contenant le code suivant :

```
class SubdomainValidator <
ActiveModel::EachValidator
  def validate_each(object, attribute, value)
    return unless value.present?

    reserved_names = %w(www ftp mail pop smtp
admin ssl sftp)
    reserved_names = options[:reserved] if
options[:reserved]

    if reserved_names.include?(value)
      object.errors[attribute] << 'cannot be a
reserved name'
    end

    object.errors[attribute] << 'must have
between 3 and 63 letters' unless (3..63) ===
value.length
    object.errors[attribute] << 'cannot start or
end with a hyphen' unless value =~ /^[^-].*[^-]
$/i
    object.errors[attribute] << 'must be
alphanumeric; A-Z, 0-9 or hyphen' unless value =~
/^[a-z0-9-]*$/i
  end
end
```

4. Gestion des cookies

L'une des dernières choses à gérer avec les sous-domaines concerne les cookies. Par défaut, les cookies concernent un domaine et un sous-domaine. Si l'on veut que les cookies soient disponibles quel que soit le sous-domaine il faut modifier l'initializer qui gère ces cookies :

```
# /config/initializers/session_store.rb

# without subdomain
# MyApp::Application.config.session_store
:cookie_store, key: '_myapp_session'

# with subdomain
```

```
MyApp::Application.config.session_store
:cookie_store, key: '_myapp_session', domain:
:all
```

5. Gestion des langues avec les sous-domaines

Hormis le fait de récupérer des objets grâce aux sous-domaines, il est possible de gérer les langues de ces derniers.

```
MyProject::Application.routes.draw do
  constraints(Subdomain) do
    root :to => 'home#index'
  end
end
```

Dans ce cas, nous allons donc avoir des URL du type fr.myapp.dev ou en.myapp.dev.
Ensuite, dans votre ApplicationController, vous pouvez récupérer la locale passée afin de définir la locale de l'application :

```
class ApplicationController <
ActionController::Base
  before_filter :set_locale

  protected
    def set_locale
      I18n.locale = request.subdomain if
request.subdomain
    end
end
```

Le principe est exactement le même que précédemment sauf que dans ce cas c'est la locale qui est passée en sous-domaine au lieu d'un identifiant permettant de récupérer un objet.

Retrouvez l'article de Synbioz en ligne : Lien 26

Créer une galerie d'images Polaroid - Uniquement en CSS

Dans ce tutoriel nous allons voir comment créer une galerie d'images Polaroid en utilisant uniquement du CSS.

1. Traduction

Ce tutoriel est la traduction la plus fidèle possible du tutoriel original de Paul Underwood, Create Polaroid Images With CSS : Lien 27.

2. Introduction

Une image Polaroid est une icône représentant une image. Cela peut paraître étrange de vouloir afficher des images Polaroid sur un écran d'ordinateur, cependant ça peut être un moyen efficace de créer une galerie d'images.
Nous allons donc nous intéresser aux images Polaroid dans ce tutoriel. Nous allons apprendre à créer une galerie de ces images uniquement grâce au CSS.

3. Le code HTML

Pour commencer, nous allons créer le code HTML qui contiendra les images Polaroid. Celui-ci est très simple, il contient seulement les images qui seront affichées dans une balise <a> afin d'avoir un lien sur celles-ci. Nous allons préciser l'attribut title pour les ancres, car ce texte sera associé à chaque image.

```
<div class="polaroid-images">
  <a href="" title="Cave"><img height="200"
src="images/water.jpg" alt="Cave" title="Cave"
/></a>
  <a href="" title="Island"><img height="200"
src="images/water2.jpg" alt="Island"
title="Island" /></a>
  <a href="" title="Islands Forest"><img
height="200" src="images/water3.jpg" alt="Islands
Forest" title="Islands Forest" /></a>
  <a href="" title="Decking"><img height="200"
src="images/water4.jpg" alt="Decking"
title="Decking" /></a>
  <a href="" title="Lake"><img height="200"
src="images/water5.jpg" alt="Lake"
title="Lake" /></a>
  <a href="" title="Mountains"><img height="200"
src="images/water6.jpg" alt="Mountains"
title="Mountains" /></a>
  <a href="" title="Forest"><img height="200"
src="images/water7.jpg" alt="Forest"
title="Forest" /></a>
  <a href="" title="Coast Valley"><img
height="200" src="images/water8.jpg" alt="Coast
Valley" title="Coast Valley" /></a>
</div>
```

4. Le code CSS

4.1. Le style Polaroid

Pour transformer nos images en Polaroid, nous allons appliquer un style particulier aux liens contenant ces dernières. Et c'est grâce à ce style que nous allons pouvoir utiliser le contenu de l'attribut title comme légende de nos images. Nous allons donc appliquer un arrière-plan blanc et un padding à nos liens pour donner l'effet Polaroid.
Le padding en dessous de l'image doit être légèrement plus grand que sur les autres côtés, car c'est à cet endroit qu'on affichera notre texte.

```
.polaroid-images a
{
        background: white;
        display: inline;
        float: left;
        margin: 0 15px 30px;
        padding: 10px 10px 25px;
        text-align: center;
        text-decoration: none;
        -webkit-box-shadow: 0 4px 6px rgba(0, 0,
0, .3);
        -moz-box-shadow: 0 4px 6px
rgba(0,0,0,.3);
        box-shadow: 0 4px 6px rgba(0,0,0,.3);
        -webkit-transition: all .15s linear;
        -moz-transition: all .15s linear;
        transition: all .15s linear;
        z-index:0;
}
```

À ce stade nous n'avons pas encore de texte affiché sous les images. Nous allons donc utiliser la pseudoclasse CSS :after afin d'ajouter un nouvel élément après nos liens. L'avantage de cette méthode est que nous allons pouvoir récupérer le contenu de l'attribut title de nos liens avec la propriété CSS content.

```
.polaroid-images a:after {
        color: #333;
        font-size: 20px;
        content: attr(title);
        position: relative;
        top:15px;
}
```

Enfin on s'assure que quelle que soit l'image elle rentrera toujours dans notre zone HTML.

```
.polaroid-images img {
        display: block;
        width: inherit;
}
```

4.2. Tourner les images

Afficher des images qui ressemblent à des Polaroid n'est pas suffisant, il va falloir ajouter un peu d'interactions sur

ces images. Nous pouvons faire en sorte que les images aient l'air d'avoir été jetées, comme on jette des Polaroid sur une table pour les regarder. Pour créer cet effet, nous allons tourner légèrement chaque image de manière aléatoire.
Pour obtenir une rotation aléatoire, nous allons utiliser le sélecteur CSS nth-child pour atteindre toutes les images et les tourner en fonction de l'ordre avec lequel elles apparaissent sur la page.

```
.polaroid-images a:nth-child(2n)
{
        -webkit-transform: rotate(4deg);
        -moz-transform: rotate(4deg);
        transform: rotate(4deg);
}
.polaroid-images a:nth-child(3n) {
        -webkit-transform: rotate(-24deg);
        -moz-transform: rotate(-24deg);
        transform: rotate(-24deg);
}
.polaroid-images a:nth-child(4n)
{
        -webkit-transform: rotate(14deg);
        -moz-transform: rotate(14deg);
        transform: rotate(14deg);
}
.polaroid-images a:nth-child(5n)
{
        -webkit-transform: rotate(-18deg);
        -moz-transform: rotate(-18deg);
        transform: rotate(-18deg);
}
```

Nous pouvons ensuite créer un effet de « ramassage » des Polaroid en utilisant l'événement hover de la souris. Comme les images seront légèrement tournées, au survol de la souris on les redimensionnera et on annulera la rotation pour les remettre droites.
On redéfinit donc la rotation à 0, l'échelle à 120% et on ajoute une ombre pour donner l'impression que l'image s'élève au-dessus de la page.

```
.polaroid-images a:hover{
        -webkit-transform: rotate(0deg);
        -moz-transform: rotate(0deg);
        transform: rotate(0deg);
        -webkit-transform: scale(1.2);
        -moz-transform: scale(1.2);
        transform: scale(1.2);
        z-index:10;
        -webkit-box-shadow: 0 10px 20px rgba(0,
0, 0, .7);
        -moz-box-shadow: 0 10px 20px
rgba(0,0,0,.7);
        box-shadow: 0 10px 20px rgba(0,0,0,.7);
}
```

Et c'est tout ce dont nous avions besoin pour créer notre galerie !

5. Conclusion

Nous avons vu dans ce tutoriel comment créer une galerie d'images façon Polaroid et ce uniquement avec du code CSS (et un peu de HTML bien évidemment). Vous pouvez vous amuser à essayer d'autres effets pour vos galeries d'images !

6. Liens

Vous pouvez consulter la démonstration pour avoir un aperçu du rendu dans un navigateur : Lien 28.

Retrouvez l'article de Paul Underwood traduit par Sébastien Germez en ligne : Lien 29.

Créer une fenêtre modale avec CSS 3

Comme beaucoup de fonctionnalités liées à des actions de l'utilisateur, les fenêtres modales étaient jusqu'à présent l'apanage de JavaScript.
Avec CSS 3, de nombreuses fonctionnalités sont désormais accessibles uniquement grâce aux feuilles de style.

1. Qu'est-ce qu'une fenêtre modale ?

Cet article met en œuvre des propriétés CSS 3. Les exemples contenus dans l'article ne sont donc fonctionnels que sur des navigateurs récents implémentant les propriétés utilisées.
En particulier, les exemples ne sont pas compatibles avec Internet Explorer 8 et inférieurs.

Pendant longtemps, les sites Web avertissaient les utilisateurs d'informations jugées importantes à l'aide d'alertes JavaScript ou de *pop-ups*.
Malheureusement, l'utilisation de ces « fenêtres surgissantes » a rapidement été associée à de mauvaises pratiques. Souvent à raison du reste : souvenez-vous de ces vieux sites où votre arrivée entraînait une multitude d'ouvertures de fenêtres non désirées.
Ces pratiques se font de plus en plus rares, car les utilisateurs et les navigateurs savent en empêcher l'utilisation.
Malgré tout, le principe de base reste potentiellement valable : bloquer l'utilisateur tant qu'il n'a pas validé une information importante, pris connaissance d'une donnée, validé un accord, fermé une image dont il a demandé l'agrandissement ou autres.
Une fenêtre modale est donc une information affichée en premier plan de la page et empêchant toute interaction avec le reste de celle-ci tant qu'une des actions prédéfinies n'a pas été accomplie.

2. Comment créer une fenêtre modale ?

Les fenêtres de type alert() en JavaScript où les *pop-ups* ne sont plus que rarement utilisées, car elles peuvent être facilement bloquées par l'utilisateur et surtout (lorsqu'elles sont justifiées) parce qu'on ne peut pas leur affecter de style.
La technique souvent utilisée pour les remplacer est donc

de couvrir la page d'un masque opaque (en semi-transparence) et d'ajouter par-dessus un élément HTML qui devient alors le seul avec lequel l'utilisateur peut interagir du fait du masque.
Jusqu'à présent, le moyen le plus simple (le seul en fait) pour mettre cela en place était l'utilisation de JavaScript, seul composant de la page Web pouvant réagir à un événement.
Mais grâce à CSS 3, il existe désormais un autre moyen de procéder, grâce à la pseudoclasse :target.

2.1. Qu'est-ce qu'une pseudoclasse ?

CSS permet de cibler des éléments particuliers d'un document à l'aide de sélecteurs. Parmi les plus connus, element permet de cibler les balises <element>, .classe permet de cibler les balises ayant la classe donnée, de même pour #id avec un identifiant.
Bien sûr, de nombreux types de sélecteurs permettent d'affiner au mieux le ciblage.
Mais CSS permet aussi de cibler des parties d'un élément en fonction de leur nature. Ce sont les pseudoéléments. C'est-à-dire qu'ils n'ont pas besoin d'être entourés de balise pour pouvoir être récupérés. Par exemple, :first-letter ou :first-line permettent de donner un style à la première lettre ou la première ligne d'un élément sans pour autant avoir besoin d'utiliser une balise pour les isoler.
De même, il est possible de cibler un élément par rapport à une interaction, il s'agit alors de pseudoclasse. Un peu comme si l'on affectait une classe spécifique dans un contexte particulier.
Parmi les pseudoclasses connues, celles concernant les liens (:link, :visited, :hover, :focus, :active) sont couramment employées et se comportent comme si le lien possédait une classe différente liée à chacun de ces états.
Avec CSS 3, il existe de nombreuses pseudoclasses. Celle qui nous intéresse dans le cadre d'une fenêtre modale étant :target.

3. La pseudoclasse :target

On peut rencontrer différents types de liens dans une page Web. Bien entendu, les liens hypertextes permettent de naviguer d'une page à une autre. D'autres liens peuvent servir à envoyer un e-mail. Enfin, certains servent à naviguer dans la même page, ce sont les ancres.
Les liens de type ancres se reconnaissent au caractère '#'. Ce qui suit ce caractère est un indicateur de l'endroit où se positionner dans la page : l'ancre (qui correspond à l'attribut id de l'élément recherché).
C'est précisément l'élément correspondant à cette ancre qui est ciblé par la pseudoclasse :target. Ainsi, lorsque l'on clique sur une ancre, il est possible de donner un style particulier à l'élément ciblé.

3.1. Exemple d'utilisation

Le cadre gris ci-dessous possède l'identifiant monAncreCible. Le premier des liens pointe vers cette ancre alors que le second lien pointe vers le premier.
Voici le code :

```
<style>
#monAncreCible{
    width: 80%;
    height: 50px;
    background-color: silver;
}
#monAncreCible:target{
    background-color: gold;
}
</style>
<div id="monAncreCible"></div>
<p>
    <a href="#monAncreCible">Changer la couleur de la boîte</a><br />
    <a href="#noWhere">Rétablir la couleur</a>
</p>
```

Comme vous pouvez le tester, lorsque vous cliquez sur le premier lien (Lien 30), la boîte change de couleur : l'élément ciblé par le lien prend bien le style déclaré avec :target. En revanche, lorsque vous cliquez sur le second lien, l'élément visé change, la boîte n'est plus affectée par la pseudoclasse et reprend son aspect d'origine.
Pour éviter que la page ne se repositionne au niveau de l'ancre lorsque l'on veut retirer l'effet de :target, on fait pointer le lien vers une ancre inexistante, comme dans l'exemple.

4. Création de la fenêtre modale

Nous savons comment utiliser la pseudoclasse :target pour interagir au clic sur un lien avec un autre élément.
Nous allons donc pouvoir utiliser cette technique pour créer une fenêtre modale. La première étape étant d'afficher le masque opaque puis d'y ajouter le message à afficher.

4.1. Création du masque

Le masque opaque ne doit être affiché que lorsque nous cliquons sur le lien souhaité. Il sera donc invisible par défaut.
De plus, nous souhaitons qu'il empêche toute interaction avec le reste de la page, il faut donc qu'il couvre totalement celle-ci.
La meilleure solution pour cela est de lui donner un display: none; et une position: fixed.
Il ne reste plus ensuite qu'à définir ses dimensions, sa couleur de fond, son opacité et si besoin son z-index.
Voici le code correspondant.

```
#overlay{
    display: none;
    position: fixed;
    top:0; right:0; bottom:0; left:0;
    background-color: black;
    opacity: 0.5;
    z-index: 1000;
}
#overlay:target{
    display: block;
}
#overlay a{
    font-weight: bold;
    padding: 10px 25px;
    background-color: white;
    position: absolute;
    top: 50%;
    left: 50px;
}
```

```
<div id="overlay"><a href="#noWhere">Supprimer le masque</a></div>
<p><a href="#overlay">Afficher le masque</a></p>
```

Notez aussi le lien ajouté par-dessus le masque : c'est le seul moyen de le faire disparaître ! Le lien est positionné sous la table des matières afin de pouvoir être visible.

Cette première étape est réussie : on arrive à afficher et masquer le calque opaque grâce aux liens et à leurs ancres. Néanmoins, on commence à comprendre les problèmes auxquels on va être confrontés concernant le contenu.

4.2. Gérer le contenu de la fenêtre

On se rend compte avec l'exemple précédent que l'on va être confronté à un problème conceptuel.
En effet, étant donné que l'apparition de la fenêtre modale se fait lors du clic sur un lien, le message à afficher devra être à l'intérieur du balisage du masque, car un lien ne peut avoir qu'une seule cible. Or l'opacité d'un élément est transmise à tous ses enfants. Ce qui fait que le message sera lui aussi opaque, ce qui n'est pas acceptable en termes d'ergonomie.
En JavaScript, la solution est simple : on crée deux éléments distincts, le masque et la fenêtre. On positionne le masque par-dessus la page puis la fenêtre par-dessus le masque, ainsi, la fenêtre contenant le message n'étant pas enfant du masque, elle n'en hérite pas la transparence.
Nous avons vu que cette façon de procéder n'est pas réalisable en CSS car il est impossible de faire afficher deux éléments distincts avec :target.
Là encore, CSS 3 va nous permettre de contourner le problème.
Si l'opacité est transmise aux balises enfants, nous allons donc la gérer au niveau de la couleur de fond en utilisant le mode rgba.
Le mode rgba est une façon de définir une couleur apparue avec CSS 3. Elle reprend ce qui existait avec rgb (*red, green, blue*) en ajoutant un paramètre *alpha* correspondant à la transparence.
Pour plus de détails, vous pouvez consulter l'article de Jérôme Debray : Comprendre les couleurs en CSS3 (Lien 31).
L'exemple précédent devient alors :

```
#overlay2{
    display: none;
    position: fixed;
    top:0; right:0; bottom:0; left:0;
    background-color: rgba(0, 0, 0, 0.5);
    z-index: 1000;
}
#overlay2:target{
    display: block;
}
#overlay2 a{
    font-weight: bold;
    padding: 10px 25px;
    background-color: white;
    position: absolute;
    top: 25%;
    left: 50px;
}
```

```
<div id="overlay2"><a href="#noWhere">Supprimer le masque</a></div>
<p><a href="#overlay2">Afficher le masque</a></p>
```

On constate que maintenant, le masque est toujours opaque, mais que son contenu est correctement affiché.
Il ne reste donc plus qu'à donner un peu de style à la fenêtre contenant le message.
Le style de la fenêtre est repris de l'article Créez une fenêtre modale avec CSS et jQuery : Lien 32.

```
#overlay3{
    display: none;
    position: fixed;
    top:0; right:0; bottom:0; left:0;
    background-color: rgba(0, 0, 0, 0.5);
    z-index: 1000;
}
#overlay3:target{
    display: block;
}
.popup_block{
    background: #fff;
    padding: 20px;
    border: 20px solid #ddd;
    position: relative;
    margin: 10% auto;
    width: 40%;
    box-shadow: 0px 0px 20px #000;
    border-radius: 10px;
}
img.btn_close {
    float: right;
    margin: -55px -55px 0 0;
}
```

```
<div id="overlay3">
    <div class="popup_block">
        <a class="close" href="#noWhere"><img
alt="Fermer" title="Fermer la fenêtre"
class="btn_close"
src="./images/close_pop.png"></a>
        <img style="float: right; margin: 0 0 0
20px;" alt="Lil bomb dude"
src="./images/bomber.gif">
        <h2>Popup</h2>

        <p>Aliquip transverbero loquor esse ille
vulputate exerci veniam fatua eros similis illum
valde. Praesent, venio conventio rusticus
antehabeo lenis. Melior pertineo feugait,
praesent hos rusticus et haero facilisis abluo.
</p>
        <p>Veniam tincidunt augue abluo vero,
augue nisl melior quidem secundum nobis
singularis eum eum.</p>

    </div>
</div>
<p><a href="#overlay3">Afficher le masque</a></p>
```

5. Autre méthode

L'utilisation de la pseudoclasse :target me semble la plus simple à mettre en place et la plus élégante. Cependant, elle présente l'inconvénient de modifier l'URL de la page, ce qui n'est pas nécessairement souhaité.
Une autre technique, un peu plus complexe et contraignante, permet de pallier ce problème.

Lorsque l'on dit que la modification de l'URL n'est pas nécessairement souhaitée, il ne s'agit pas de considération esthétique concernant la barre d'adresse du navigateur.
Dans certaines applications Web, la partie hash de l'URL (celle qui se trouve après le signe #) est utilisée pour conserver l'état de l'affichage (ce que l'on appelle le *hashtag* ou *hashbang*). Dans ce cas, la modification de l'URL devient réellement problématique.

5.1. La pseudoclasse :checked

La pseudoclasse :checked permet de cibler des boutons radio ou des cases à cocher en fonction de leur état.
Son utilisation peut être visualisée dans l'exemple suivant :

```
.checkbox:checked{
    box-shadow: 0 0 10px 2px gold;
}
```

Vous pouvez constater qu'en cochant ou décochant la case, son contour est modifié.
En tant que telle, cette pseudoclasse ne nous sera pas particulièrement utile puisqu'elle ne permet pas de cibler d'autres éléments que les balises <input /> correspondantes. Il va donc falloir la coupler avec un autre sélecteur pour pouvoir s'en servir efficacement pour notre fenêtre modale.
En règle générale, CSS permet de cibler un élément par rapport à ses parents successifs : la recherche se fait en avançant progressivement dans l'arborescence HTML. Ce qui ne nous est pas très utile puisqu'une balise <input /> ne peut pas avoir d'enfant.
Heureusement, il existe malgré tout certains sélecteurs permettant la recherche sur le même niveau. Dans cet exemple, nous utiliserons le sélecteur de frère adjacent (Lien 33) E + F.
Ce sélecteur permet de cibler tout élément F frère (c'est-à-dire au même niveau de l'arborescence, ou ayant le même parent) de l'élément E.

Cette particularité va avoir pour conséquence d'imposer une structure HTML figée, alors que l'exemple précédent est plus souple sur ce point.

À partir de ces informations, nous allons pouvoir élaborer une ébauche de structure pour notre fenêtre modale.

```
#overlay4{
    display: none;
    position: fixed;
    top: 0;
    bottom: 0;
    left: 0;
    right: 0;
    background-color: rgba(0, 0, 0, 0.5);
    z-index: 1000;
}
#modalCheck4{
    position: relative;
    z-index: 1001;
}
#modalCheck4:checked + #overlay4{
    display: block;
}
```

```
<div>
    <label for="modalCheck4">Cliquer pour
afficher le masque : </label><input
type="checkbox" id="modalCheck4" />
    <div id="overlay4">
    </div>
</div>
```

Dans cet exemple, nous avons rajouté un positionnement et un z-index à la case à cocher afin de la mettre au premier plan et de permettre de la décocher.

5.2. Gérer le contenu de la fenêtre

Pour que l'exemple précédent soit opérationnel, nous sommes obligés de positionner la case à cocher en premier plan pour permettre de la décocher. Or dans l'absolu, c'est plutôt le contraire que l'on souhaiterait faire : que la case ne soit jamais visible !
Nous allons contourner cette obligation en utilisant la balise <label>, qui permet d'associer son contenu à la balise <input /> et de cocher cette dernière lorsque l'on clique dessus, ce qui va nous permettre de masquer la case à cocher.
D'autre part, plusieurs balises <label> peuvent être associées au même élément.
Nous adaptons donc notre exemple pour prendre en compte ces informations :

```
#overlay5{
    display: none;
    position: fixed;
    top: 0;
    bottom: 0;
    left: 0;
    right: 0;
    background-color: rgba(0, 0, 0, 0.5);
    z-index: 1000;
}
#modalCheck5{
    display: none;
}
#modalCheck5:checked + #overlay5{
    display: block;
}
label{
    cursor: pointer;
}
.labelButton{
    border: 1px solid black;
    padding: 5px 10px;
    background-color: white;
    position: absolute;
    top: 25%;
    left: 50px;
}
```

```
<div>
    <label class="lienArticle simple"
for="modalCheck5">Cliquer pour afficher le
masque</label><input type="checkbox"
id="modalCheck5" />
    <div id="overlay5">
        <label class="labelButton"
for="modalCheck5">Cliquer pour supprimer le
masque</label>
    </div>
</div>
```

Le résultat correspond à nos attentes. Il ne reste plus qu'à ajouter du contenu à la fenêtre modale avec le même code que dans la partie précédente.

```
#overlay6{
    display: none;
    position: fixed;
    top: 0;
    bottom: 0;
    left: 0;
    right: 0;
    background-color: rgba(0, 0, 0, 0.5);
    z-index: 1000;
}
#modalCheck6{
    display: none;
}
#modalCheck6:checked + #overlay6{
    display: block;
}
```

```
<div>
    <label for="modalCheck6">Voir la fenêtre
modale</label><input type="checkbox"
id="modalCheck6" />
    <div id="overlay6">
        <div class="popup_block">
            <label for="modalCheck6"><img
alt="Fermer" title="Fermer la fenêtre"
class="my_btn_close" src="./images/close_pop.png"
/></label>
            <img style="float: right; margin: 0 0
0 20px;" alt="Lil bomb dude"
src="./images/bomber.gif" />
            <h2>Popup</h2>
            <p>Aliquip transverbero loquor esse
ille vulputate exerci veniam fatua eros similis
illum valde. Praesent, venio conventio rusticus
antehabeo lenis. Melior pertineo feugait,
praesent hos rusticus et haero facilisis abluo.
</p>
            <p>Veniam tincidunt augue abluo vero,
augue nisl melior quidem secundum nobis
singularis eum eum.</p>

        </div>
    </div>
</div>
```

Attention, cette technique possède les mêmes contraintes que la précédente en termes de compatibilité. Elle ne fonctionnera pas avec les versions d'Internet Explorer inférieures à 9.

6. Conclusion

Vous pouvez voir un exemple en ligne (Lien 34) ou télécharger l'archive (Lien 35).
Bien entendu, il ne s'agit dans cet article que de montrer la façon de procéder. Il est possible d'ajuster le code pour qu'il corresponde mieux à vos besoins.
D'autre part, CSS 3 permet beaucoup d'autres améliorations, notamment faire apparaître et disparaître la fenêtre avec des animations ou des transformations.
Enfin, gardez à l'esprit que malgré la prise en charge croissante de CSS 3 par les navigateurs, les codes présentés dans cet article ne sont compatibles pour Internet Explorer qu'à partir de la version 9.

Retrouvez l'article de Didier Mouronval en ligne : Lien 36

Dimensionner son texte avec rem - Une nouvelle unité disponible en CSS3

Dans cet article, nous allons apprendre à utiliser une nouvelle unité pour dimensionner son texte : le rem.

1. Introduction

Le dimensionnement du texte sur une page Web est, encore aujourd'hui, un sujet épineux. La seule variable commune à tous les développeurs est l'utilisation de la propriété CSS font-size.
Maintenant une question se pose : quelle unité utiliser avec cette propriété ?
Doit-on préférer les px ? Les em ? Les pourcentages ? Les valeurs absolues qui correspondent aux anciennes tailles de texte HTML (allant de 1 à 7) telles que small, large ou encore xx-large ? Ou bien carrément utiliser une des unités dites absolues proposées par le W3C : pt (point), pc (pica = 12 points), cm (centimètre), mm (millimètre) ou in (pouce, inch en anglais). Ces dernières étant plutôt destinées à de l'impression ; en effet, elles varient d'un écran à l'autre en fonction de la résolution de celui-ci, il serait donc très compliqué d'adapter l'apparence pour différents appareils.
On s'orientera donc plus vers les unités relatives que sont les px, les em et les pourcentages. Chacune ayant ses avantages et ses inconvénients que je détaillerai brièvement dans le chapitre suivant.
Cependant, une nouvelle unité a fait son apparition avec la version 3 des CSS : rem. Je vais vous expliquer son fonctionnement et vous présenter un cas d'utilisation.

2. Unités de dimensionnement avant CSS3

2.1. Le dimensionnement en px

Cette façon de dimensionner le texte est la toute première apparue sur le Web. C'était une méthode simple et fiable à l'époque où tout un chacun était équipé d'un écran avec une résolution de 800x600. Cependant, à cette époque l'expérience utilisateur était peu prise en compte et l'utilisation des pixels présente plusieurs inconvénients :

- aujourd'hui les résolutions diffèrent énormément, surtout avec l'apparition des appareils mobiles (smartphone, tablette). On ne peut donc pas fixer la taille de la police en pixels et espérer avoir un design adapté à différents appareils ;
- le texte ne peut être agrandi par les fonctionnalités natives du navigateur.

Vous l'aurez compris, le dimensionnement en pixels reste très rigide et peu adapté au mouvement actuel qui consiste à faire du responsive design. Pour rappel, le responsive design est un principe dont l'importance croît avec l'évolution des smartphones et tablettes. En effet, ce principe consiste à faire en sorte que son site Web s'affiche correctement, quel que soit le type d'appareil utilisé pour

le visionner. Il s'agit donc d'adapter son design aussi bien à un ordinateur de bureau qu'à une tablette ou encore un ordinateur portable. Il existe différentes méthodes pour y parvenir, je vous invite à consulter la page Cours CSS pour en savoir plus : Lien 37.

2.2. Le dimensionnement en pourcentage

La première alternative aux pixels est l'utilisation de pourcentage pour optimiser l'affichage. On décide que la barre de menu occupera 30 % de la page, le contenu 60 %, etc.
Cependant, on se heurte de nouveau à un inconvénient lié aux résolutions actuelles. Ainsi, sur de très grandes résolutions, un site dont l'affichage est géré en pourcentage sera très étiré, là où sur une très petite résolution le site s'affichera complètement écrasé.

2.3. Le dimensionnement en em

La troisième possibilité pour dimensionner du texte est d'utiliser l'unité em. Pour la petite histoire, cette unité est appelée cadratin en français. Elle correspond à la hauteur d'un caractère d'imprimerie en plomb. Celle-ci résout le problème lié à l'agrandissement du texte via les fonctionnalités de chaque navigateur.
Grâce à cette unité, on peut en effet dimensionner notre texte en fonction de la taille de texte de son parent. La définition étant obscure, rien ne vaut une explication par l'exemple. Supposons le code suivant :

```
body { font-size : 100% ; }
h1 { font-size : 3em ; }
```

Dans notre exemple, le texte contenu dans un titre de niveau 1 sera trois fois plus grand que le texte du corps de la page. Vous allez me dire que c'est la solution miracle qui résout tous les problèmes ! Et je vais vous répondre : non.
En effet, il existe un énorme inconvénient au dimensionnement en em : les tailles se répandent en cascade. Imbriquez plusieurs listes (<li>) pour lesquelles vous définissez la même taille en em et vous vous apercevrez que le texte n'a pas du tout la même taille dans chaque liste. Eh oui, car cette méthode se réfère au parent direct et il est impossible de faire un reset du contexte pour préciser qu'on utilise la taille de texte du body en tant que parent. Si on suppose le code suivant :

```
body { font-size : 100% ; }
li { font-size : 1.5em ; }
```

Le texte de notre deuxième liste imbriquée dans la première aura une taille 1,5 fois plus grande que… le texte de la première liste ! Je vous laisse imaginer les problèmes que cela peut engendrer lors d'imbrication sur plusieurs niveaux…
C'est précisément à ce niveau qu'intervient l'unité rem.

3. Le dimensionnement en rem

CSS3 est arrivé avec son lot de nouveautés et bien heureusement pour nous avec une nouvelle unité appelée rem. Je vous vois venir d'ici me dire que rien ne change par rapport au cadratin. Eh bien si ! Comme son nom l'indique, rem signifie root em (em à la racine). Cette unité ne se base plus sur le parent direct pour calculer les tailles de texte mais sur la taille de texte de la racine de la page Web à savoir l'élément <html>. Et c'est précisément cette petite différence qui va résoudre le problème de cascade !
Nous pouvons dorénavant définir une taille de texte par défaut sur notre élément racine <html> et utiliser l'unité rem pour tous les autres éléments enfants afin de calculer une taille de texte proportionnelle. N'importe quelle taille spécifiée en rem sera calculée sur la base de la taille de l'élément racine : fini les problèmes de taille de texte en cascade !
Reprenons notre exemple de liste :

```
html { font-size : 100% ; }
li { font-size : 1.5rem ; }
```

Avec ce code, le texte de notre deuxième liste aura exactement la même taille que celui de la première liste, car quel que soit le niveau d'imbrication, la taille du texte de nos listes sera 1,5 fois plus grande que le texte de notre élément racine à savoir l'élément <html>.

4. Support des navigateurs

Un dernier point pourrait poser problème : le support des navigateurs. En effet, l'unité rem a fait son apparition dans la version 3 de CSS. On peut donc légitimement se demander quels navigateurs supportent cette méthode. Aussi surprenant que cela puisse paraître, le support de cette unité est plutôt bon :

- Chrome toutes versions ;
- Firefox 3.6+ ;
- Internet Explorer 9 ;
- Safari 5+ ;
- Opera 12.1+.

Malgré ce support étonnamment bon, la plupart des développeurs d'aujourd'hui se soucient de **tous** les utilisateurs. Quelle solution adopter dans ce cas pour les utilisateurs d'anciennes versions d'Internet Explorer par exemple comme c'est le cas dans beaucoup de grosses entreprises ?
Nous pouvons simplement coupler une taille de texte en pixels à celle en rem. Les utilisateurs dont le navigateur prendra en compte la taille en pixels ne pourront pas redimensionner leur texte via leur navigateur, mais c'est mieux que rien.
En reprenant notre exemple de liste cela donnerait :

```
html { font-size : 100% ; }
li {
  font-size : 24px ;
  font-size : 1.5rem ;
}
```

Si vous vous demandez pourquoi 24, la réponse est simple : la taille de texte par défaut est fixée à 16px. Comme nous définissons notre taille de base à 100 % nous ne modifions donc pas la taille par défaut, le texte par défaut sera donc également de 16px dans notre cas. Il suffit donc ensuite de calculer la taille de texte pour nos listes à partir de cette valeur ce qui nous donne une équivalence en pixels de 24 pour une taille de 1.5em.

5. Conclusion

J'espère avoir été clair sur l'utilisation de cette nouvelle

unité introduite en CSS3 qu'est le rem. Rien ne vous empêche de définir la taille de base de votre élément racine en pixels. Cependant, on perd l'intérêt des unités relatives. Chacun optera pour la solution qui lui parle le plus, mais vous devez bien garder en tête qu'avec l'évolution croissante du marché du mobile et des tablettes le responsive design devient de plus en plus important. On ne peut négliger aujourd'hui les utilisateurs de ces appareils si l'on veut rester compétitif !

Retrouvez l'article de Sébastien Germez en ligne : Lien 38

L'API HTML5 de contrainte de validation - La validation native des formulaires côté client

Cet article est la traduction de Constraint Validation: Native Client Side Validation for Web Forms publié sur le site HTML5 Rocks : Lien 39.

1. Introduction

La validation des formulaires a longtemps été une expérience pénible de développement. Implémenter une validation côté client qui soit à la fois conviviale pour l'utilisateur, souple pour le développeur et accessible est compliqué. Avant HTML5, il n'existait aucune façon d'obtenir nativement une validation et les développeurs devaient passer par différentes solutions basées sur JavaScript.
Afin de soulager le développeur de ce fardeau, HTML5 a introduit le concept connu sous le nom de contrainte de validation : Lien 40. Une solution native pour implémenter la validation des formulaires Web côté client.
Actuellement, bien que disponible sur les dernières versions de tous les navigateurs, l'API de contrainte de validation est reléguée aux présentations et démonstrations. Mon but dans cet article est d'éclairer les développeurs sur les possibilités de cette nouvelle API et leur permettre de créer de meilleurs formulaires pour tout le monde.
Dans cet article, nous allons :

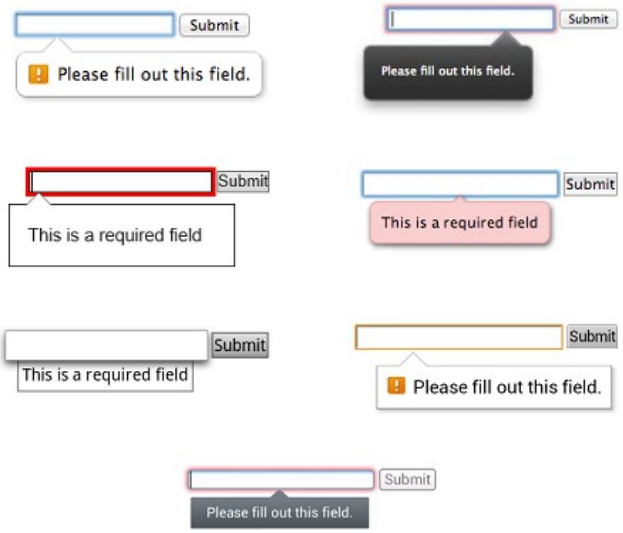
- présenter une vue d'ensemble détaillée de ce qu'est la contrainte de validation ;
- faire le point sur les limites actuelles au niveau de la spécification et des différentes implémentations ;
- voir comment utiliser la contrainte de validation HTML5 dans vos formulaires actuellement.

2. Qu'est-ce que la contrainte de validation ?

Le cœur de l'API de validation est un algorithme lancé par le navigateur lorsqu'un formulaire est soumis et permettant de déterminer si les valeurs qu'il contient sont valides. Pour effectuer cette vérification, l'algorithme utilise les nouveaux attributs HTML5 min (Lien 41), max (Lien 42), step (Lien 43), pattern (Lien 44) et required (Lien 45) ainsi que les attributs existants maxlength (Lien 46) et type (Lien 47).
À titre d'exemple, prenez ce formulaire avec un champ texte et un attribut required :

```
<form>
    <input type="text" required value="" />
    <input type="submit" value="Submit" />
</form>
```

Si vous essayez de soumettre ce formulaire tel quel, les navigateurs compatibles (Lien 48) vont empêcher la soumission et afficher un des messages suivants (NdT les messages ne sont pas traduits) :



Selon la spécification, la façon dont le message est présenté à l'utilisateur est laissée à l'initiative du navigateur. Par ailleurs, la spécification prévoit une API DOM complète, de nouveaux attributs HTML et des règles CSS que les auteurs peuvent utiliser pour personnaliser l'interface.

3. L'API DOM

L'API de contrainte de validation ajoute les propriétés et méthodes suivantes aux éléments du DOM : Lien 49.

3.1. willValidate

La propriété willValidate indique si un élément sera concerné ou non par la validation. Pour les éléments pouvant être soumis (Lien 50), sa valeur sera true à moins que, pour une raison quelconque, l'élément ne soit pas éligible pour la validation, par exemple s'il possède l'attribut disabled.

```
<div id="one"></div>
<input type="text" id="two" />
<input type="text" id="three" disabled />
<script>

document.getElementById('one').willValidate;
//undefined

document.getElementById('two').willValidate;
//true

document.getElementById('three').willValidate; //
false
</script>
```

3.2. validity

La propriété validity (Lien 51) d'un élément retourne un objet ValidityState (Lien 52) contenant des propriétés booléennes relatives à la validité de la valeur de l'élément.

- customError: true si un message personnalisé a été défini avec setCustomValidity().

```
<input id="foo" />
<input id="bar" />
<script>

document.getElementById('foo').validity.customErr
or; //false

document.getElementById('bar').setCustomValidity(
'Invalid');

document.getElementById('bar').validity.customErr
or; //true
</script>
```

- patternMismatch: true si la propriété value n'est pas conforme à l'attribut pattern.

```
<input id="foo" pattern="[0-9]{4}"
value="1234" />
<input id="bar" pattern="[0-9]{4}"
value="ABCD" />
<script>

document.getElementById('foo').validity.patternMi
smatch; //false

document.getElementById('bar').validity.patternMi
smatch; //true
</script>
```

- rangeOverflow: true si la propriété value est supérieure à l'attribut max.

```
<input id="foo" type="number" max="2"
value="1" />
<input id="bar" type="number" max="2"
value="3" />
<script>

document.getElementById('foo').validity.rangeOver
flow; //false

document.getElementById('bar').validity.rangeOver
flow; //true
</script>
```

- rangeUnderflow: true si la propriété value est inférieure à l'attribut min.

```
<input id="foo" type="number" min="2"
value="3" />
<input id="bar" type="number" min="2"
value="1" />
<script>

document.getElementById('foo').validity.rangeUnde
rflow; //false

document.getElementById('bar').validity.rangeUnde
rflow; //true
</script>
```

- stepMismatch: true si la propriété value ne correspond pas à l'intervalle précisé par l'attribut step.

```
<input id="foo" type="number" step="2"
value="4" />
<input id="bar" type="number" step="2"
value="3" />
<script>

document.getElementById('foo').validity.stepMisma
tch; //false

document.getElementById('bar').validity.stepMisma
tch; //true
</script>
```

- tooLong: true si la propriété value est plus longue que l'attribut maxlength.

Tous les navigateurs empêchent ce cas de se produire normalement en interdisant la saisie si elle excède la valeur de maxlength. Cependant, dans de rares cas, cette propriété peut valoir true dans certains navigateurs. J'ai écrit un billet blog à ce sujet : Lien 53.

- typeMismatch: true si la propriété value ne correspond pas à l'attribut type.

```
<input id="foo" type="url" value="http://foo.com"
/>
<input id="bar" type="url" value="foo" />

<input id="foo2" type="email" value="foo@foo.com"
/>
<input id="bar2" type="email" value="bar" />

<script>

document.getElementById('foo').validity.typeMisma
tch; //false

document.getElementById('bar').validity.typeMisma
tch; //true


document.getElementById('foo2').validity.typeMism
atch; //false

document.getElementById('bar2').validity.typeMism
atch; //true
</script>
```

- valueMissing: true si l'élément possède l'attribut required et que sa propriété value est vide.

```
<input id="foo" type="text" required
value="foo" />
<input id="bar" type="text" required value="" />
<script>

document.getElementById('foo').validity.valueMiss
ing; //false

document.getElementById('bar').validity.valueMiss
ing; //true
</script>
```

- valid: true si toutes les conditions listées ci-dessus valent false.

```
<input id="valid-1" type="text" required
value="foo" />
```

```
<input id="valid-2" type="text" required value="" />
<script>
    document.getElementById('valid-1').validity.valid; //true
    document.getElementById('valid-2').validity.valid; //false
</script>
```

3.3. validationMessage

La propriété validationMessage d'un élément contient le message que le navigateur doit afficher à l'utilisateur si la validation du champ échoue.
Chaque navigateur possède un message par défaut pour la langue de l'utilisateur. Si l'élément n'est pas éligible pour la validation ou s'il contient une valeur correcte, validationMessage contiendra une chaine vide.

Au moment d'écrire cet article (NdT 17 octobre 2012), Opera ne remplit pas correctement cette propriété. Le message est bien affiché, mais la propriété n'est pas remplie correctement.

```
<input type="text" id="foo" required />
<script>

document.getElementById('foo').validationMessage;
    //Chrome  --> 'Please fill out this field.'
    //Firefox --> 'Please fill out this field.'
    //Safari  --> 'value missing'
    //IE10    --> 'This is a required field.'
    //Opera   --> ''
</script>
```

3.4. checkValidity()

La méthode checkValidity s'applique à un élément de formulaire (input, select, textarea…) et renvoie true si la valeur de cet élément est valide.
Sur les éléments form, elle renvoie true si l'ensemble des données à valider qu'il contient est valide.

```
<form id="form-1">
    <input id="input-1" type="text" required />
</form>
<form id="form-2">
    <input id="input-2" type="text" />
</form>
<script>
    document.getElementById('form-1').checkValidity();  //false
    document.getElementById('input-1').checkValidity(); //false

    document.getElementById('form-2').checkValidity();  //true
    document.getElementById('input-2').checkValidity(); //true
</script>
```

De plus, à chaque fois que la validité d'un champ est vérifiée avec checkValidity et que le test échoue, un événement invalid est déclenché pour cet élément. Avec l'exemple ci-dessus, si vous vouliez lancer une action lorsque l'élément input-1 est vérifié et est invalide, vous pourriez faire :

```
document.getElementById('input-1').addEventListener('invalid', function() {
    //...
}, false);
```

Il n'existe pas d'événement valid, mais vous pouvez utiliser l'événement change pour afficher vos notifications sur un élément :

```
document.getElementById('input-1').addEventListener('change', function(event) {
    if (event.target.validity.valid) {
        //Field contains valid data.
    } else {
        //Field contains invalid data.
    }
}, false);
```

3.5. setCustomValidity()

La méthode setCustomValidity modifie la propriété validationMessage et vous permet de modifier les règles de validation.
Puisqu'elle permet de modifier validationMessage, si vous lui affectez une chaine vide rend le champ valide alors que lui affecter une chaine non vide rend le champ invalide. Malheureusement, il n'est pas possible de modifier validationMessage sans modifier la validité du champ.
Par exemple, si vous avez deux champs mot de passe qui doivent être égaux, vous pouvez faire :

```
if (document.getElementById('password1').value != document.getElementById('password2').value) {

document.getElementById('password1').setCustomValidity('Passwords must match.');
} else {

document.getElementById('password1').setCustomValidity('');
}
```

4. Attributs HTML

Nous avons déjà vu que les attributs maxLength, min, max, step, pattern, et type sont utilisés par le navigateur pour vérifier la validité du formulaire. Pour la contrainte de validation, deux autres attributs peuvent être utilisés : novalidate et formnovalidate.

4.1. novalidate

L'attribut booléen novalidate est appliqué à un formulaire. Lorsqu'il est présent, il permet d'indiquer que ce champ n'aura pas besoin d'être vérifié lors de la soumission du formulaire.

```
<form novalidate>
    <input type="text" required />
    <input type="submit" value="Submit" />
</form>
```

Puisque le formulaire possède l'attribut novalidate, il sera soumis malgré la présence de l'attribut required sur le champ texte.

4.2. formnovalidate

L'attribut booléen formnovalidate s'applique à un élément submit de formulaire (button ou input).

```
<form>
    <input type="text" required />
    <input type="submit" value="Validate" />
    <input type="submit" value="Do NOT Validate"
formnovalidate />
</form>
```

Lorsque le bouton « Validate » est cliqué, le formulaire n'est pas soumis si le champ texte est vide. En revanche, lors du clic sur le bouton « Do NOT Validate », le formulaire est bien soumis du fait de son attribut formnovalidate.

5. Règles CSS

Créer une validation de formulaire efficace ne se limite pas à gérer les erreurs, cela consiste aussi à afficher ces erreurs de façon utile. Les navigateurs compatibles permettent l'utilisation de CSS pour améliorer cet affichage.

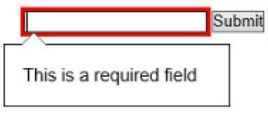
5.1. :invalid et :valid

Pour les navigateurs compatibles, la pseudoclasse :valid va cibler les éléments respectant leur règle de validation alors que la pseudoclasse :invalid ciblera ceux pour lesquels ce n'est pas le cas.

```
<form>
    <input type="text" id="foo" required />
    <input type="text" id="bar" />
</form>
<script>

document.querySelectorAll('input[type="text"]:inv
alid'); //Matches input#foo

document.querySelectorAll('input[type="text"]:val
id');   //Matches input#bar
</script>
```

5.2. Redéfinir les styles par défaut

Par défaut, Firefox applique une propriété box-shadow et IE10 une propriété outline rouge aux champs :invalid.



Par défaut, les navigateurs basés sur Webkit et Opera n'appliquent aucun style à ces éléments.
Si vous voulez un affichage similaire quel que soit le navigateur, vous pouvez supprimer les styles par défaut.

```
:invalid {
    box-shadow: none; /* FF */
    outline: 0;       /* IE 10 */
}
```

J'ai proposé une demande de *pull* (Lien 54) pour débattre de savoir si cette unification était du ressort de normalize.css (Lien 55).

5.3. Les bulles d'information

Une contradiction majeure d'affichage est le style de la bulle d'erreur affichée par les navigateurs. Webkit est le seul moteur de rendu permettant leur personnalisation. Pour Webkit, vous pouvez utiliser ces quatre pseudoclasses :

```
::-webkit-validation-bubble {}
::-webkit-validation-bubble-message {}
::-webkit-validation-bubble-arrow {}
::-webkit-validation-bubble-arrow-clipper {}
```

5.4. Supprimer l'affichage par défaut

Puisque seul Webkit permet de personnaliser le style d'affichage des erreurs, si vous souhaitez un affichage identique sur tous les navigateurs, la seule solution est de supprimer l'affichage par défaut et de créer le vôtre. Le code suivant permet de désactiver l'affichage par défaut des erreurs.

```
var forms =
document.getElementsByTagName('form');
for (var i = 0; i < forms.length; i++) {
    forms[i].addEventListener('invalid',
function(e) {
        e.preventDefault();
        // Créez votre affichage d'erreur ici.
    }, true);
}
```

Si vous désactivez les messages par défaut, faites bien attention de mettre en place votre propre affichage pour avertir les utilisateurs en cas d'erreur. Actuellement, les bulles d'information sont les seuls éléments que mettent en place les navigateurs pour prévenir les utilisateurs qu'ils se sont trompés.

6. Problèmes et limitations des implémentations

Bien que cette nouvelle API apporte beaucoup pour la validation des formulaires côté client, vous pourrez être confrontés à certaines limitations.

6.1. setCustomValidity

S'il s'agit juste de définir validationMessage sur un champ, setCustomValidity fonctionne correctement. Mais si le formulaire devient plus complexe, setCustomValidity montre des limites.

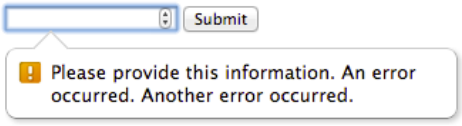
6.1.1. Gérer des erreurs multiples sur un champ

Affecter la propriété validationMessage sur un champ se contente de redéfinir validationMessage. Ainsi, si vous affectez setCustomValidity une seconde fois, vous réaffecterez juste la valeur de la propriété. Il n'y a pas de mécanisme permettant de gérer une pile d'erreurs ou de moyen d'afficher plusieurs messages.
Une solution possible pour afficher des messages d'erreurs distincts à l'utilisateur est la suivante :

```
var foo = document.getElementById('foo');
foo.setCustomValidity(foo.validationMessage + '
An error occurred');
```

Vous ne pouvez pas affecter de HTML ou de caractères de

formatage, donc le message ressemblera à ça :



6.1.2. Savoir quand vérifier la validité d'un champ

Pour illustrer ce problème, prenons le code suivant, avec un formulaire contenant deux champs password qui doivent être identiques :

```
<form>
  <fieldset>
    <legend>Change Your Password</legend>
    <ul>
      <li>
       <label for="password1">Password 1:</label>
       <input type="password" required
id="password1" />
      </li>
      <li>
       <label for="password2">Password 2:</label>
       <input type="password" required
id="password2" />
      </li>
    </ul>
  <input type="submit" />
  </fieldset>
</form>
```

Ma précédente proposition était d'utiliser l'événement change pour effectuer la vérification, ce qui peut se coder comme suit :

```
var password1 =
document.getElementById('password1');
var password2 =
document.getElementById('password2');

var checkPasswordValidity = function() {
    if (password1.value != password2.value) {
        password1.setCustomValidity('Passwords
must match.');
    } else {
        password1.setCustomValidity('');
    }
};

password1.addEventListener('change',
checkPasswordValidity, false);
password2.addEventListener('change',
checkPasswordValidity, false);
```

Avec ce code, lorsque l'un des champs est modifié, sa validité est vérifiée. Pourtant, envisageons un script qui remplit automatiquement ces champs ou encore un script qui modifie des attributs de validation comme pattern, required, min, max ou step. De tels scripts affecteraient la validité des champs, mais aucun événement ne permet de les détecter.
Nous aurions besoin d'un événement permettant de savoir quand une règle de validité a été modifiée.

6.1.3. Savoir quand l'utilisateur tente de valider un formulaire

Pourquoi ne pas utiliser l'événement submit pour le savoir ? L'événement submit n'est pas déclenché tant que le navigateur n'a pas déterminé que tous les champs à valider sont corrects. Ainsi, il n'est pas possible de savoir si l'utilisateur a tenté de valider son formulaire, mais que le navigateur l'en a empêché.
Il serait utile de savoir quand l'utilisateur tente de soumettre un formulaire. Vous pourriez vouloir montrer à l'utilisateur une liste de messages d'erreur, changer le focus ou afficher des messages d'aide. Malheureusement, vous aurez besoin de code supplémentaire pour faire cela.
L'une des façons d'y arriver est d'ajouter au formulaire l'attribut novalidate et d'utiliser l'événement submit. Du fait de l'attribut novalidate, la soumission du formulaire ne sera pas empêchée par un champ non validé. Ensuite, ce sera au script de vérifier la validité du formulaire lors de l'événement submit et d'empêcher, si besoin, la soumission.
Voici comment pourrait s'écrire un tel script dans le cas précédent de deux champs password devant être identiques :

```
<form id="passwordForm" novalidate>
    <fieldset>
        <legend>Changer votre mot de
passe</legend>
        <ul>
            <li>
                <label for="password1">Mot de
passe :</label>
                <input type="password" required
id="password1" />
            </li>
            <li>
                <label
for="password2">Confirmez :</label>
                <input type="password" required
id="password2" />
            </li>
        </ul>
        <input type="submit" />
    </fieldset>
</form>
<script>
    var password1 =
document.getElementById('password1');
    var password2 =
document.getElementById('password2');

    var checkPasswordValidity = function() {
        if (password1.value != password2.value) {
            password1.setCustomValidity('Les mots
de passe doivent être identiques.');
        } else {
            password1.setCustomValidity('');
        }
    };

    password1.addEventListener('change',
checkPasswordValidity, false);
    password2.addEventListener('change',
checkPasswordValidity, false);

    var form =
document.getElementById('passwordForm');
    form.addEventListener('submit', function() {
```

```
        checkPasswordValidity();
        if (!this.checkValidity()) {
            event.preventDefault();
            // Ajoutez ici la gestion de vos
messages d'erreur.
            password1.focus();
        }
    }, false);
</script>
```

L'inconvénient ici est que l'utilisation de l'attribut novalidate empêche l'affichage des messages par le navigateur. De ce fait, si vous utilisez cette technique, vous devez prévoir votre propre gestion d'affichage des erreurs.
Voici un exemple pour faire cela : Lien 56.
Nous aurions besoin d'un événement forminvalid qui serait déclenché lors de l'annulation d'une soumission de formulaire si les données sont invalides.

6.2. Safari

Même si Safari supporte l'API de validation, au moment de l'écriture de cet article (version 6), Safari n'empêche pas l'envoi du formulaire si les données sont invalides. Pour l'utilisateur, Safari se comporte comme un navigateur ne supportant pas l'API.
La parade la plus simple est de faire comme dans le cas précédent et d'ajouter l'attribut novalidate à tous les formulaires et de vérifier soi-même les données en empêchant la soumission avec preventDefault.
Le code suivant montre comment mettre cela en place :

```
var forms =
document.getElementsByTagName('form');
for (var i = 0; i < forms.length; i++) {
    forms[i].noValidate = true;

    forms[i].addEventListener('submit',
function(event) {
        //Prevent submission if checkValidity on
the form returns false.
        if (!event.target.checkValidity()) {
            event.preventDefault();
            // Ajoutez ici la gestion de vos
messages d'erreur.
        }
    }, false);
}
```

Il existe plusieurs bogues connus où ChackValidity renvoie des faux positifs (voir ici par exemple : Lien 57). Les faux positifs sont problématiques parce que l'utilisateur sera bloqué alors que ses données sont justes, il faut y penser !

6.3. Messages d'erreur définis

Même si vous avez la possibilité de modifier un message d'erreur avec validationMessage et setCustomValidity, cela peut être gênant de devoir obligatoirement utiliser JavaScript pour définir ces messages, surtout sur de grands formulaires.
Pour vous aider, Firefox a introduit un attribut x-moz-errormessage qui permet d'affecter directement la propriété validationMessage d'un champ.

```
<form>
    <input type="text" required x-moz-
errormessage="Fill this out." />
    <input type="submit" value="Submit" />
</form>
```

Lorsque le formulaire ci-dessus est soumis, le message d'erreur affiché par Firefox sera personnalisé :

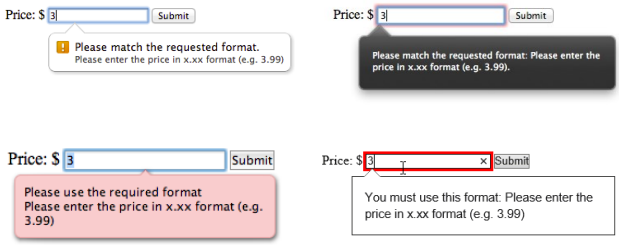
Cette fonctionnalité a été proposée au W3C (Lien 58), mais a été refusée. Pour l'instant, Firefox est le seul navigateur permettant de personnaliser de la sorte les messages d'erreur.

6.4. L'attribut title

Bien que l'attribut title ne modifie pas le message de validationMessage, les navigateurs affichent sa valeur dans la bulle d'information (s'il est présent) en cas de patternMismatch. Notons que Chrome affiche la valeur de l'attribut title quelle que soit l'erreur de validation, pas uniquement pour patternMismatch.
Par exemple, si vous essayez de soumettre le formulaire suivant :

```
<form>
    <label for="price">Price: $</label>
    <input type="text" pattern="[0-9].[0-9][0-9]"
        title="Please enter the price in x.xx
format (e.g. 3.99)"
        id="price" value="3" />
    <input type="submit" value="Submit" />
</form>
```

Voici ce que vous pourrez obtenir dans les différents navigateurs :



6.5. invalid et :valid

Comme nous l'avons déjà évoqué, la pseudoclasse :valid va cibler tous les éléments qui satisfont leurs contraintes de validation et :invalid va cibler ceux qui ne la satisfont pas. Malheureusement, ces pseudoclasses sont affectées immédiatement, c'est-à-dire avant une tentative de soumission. Prenons l'exemple suivant :

```
<style>
    :invalid {
        border: 1px solid red;
    }
```

```
    :valid {
        border: 1px solid green;
    }
</style>
<form>
    <input type="text" required />
    <input type="text" />
</form>
```

Le but ici est d'ajouter une bordure rouge sur les champs invalides et verte sur les champs valides. Mais la bordure apparaît dès l'affichage du formulaire, or des tests sur la facilité d'utilisation de formulaires (Lien 59) ont montré que le meilleur moment pour donner des informations à l'utilisateur est immédiatement **après** qu'ils ont rempli un champ, pas avant.
Une solution pour faire cela sur l'exemple précédent est d'ajouter une classe CSS aux champs après qu'ils ont été remplis et de n'appliquer les styles qu'aux éléments ayant cette classe.

```
<style>
    .interacted:invalid {
        border: 1px solid red;
    }
    .interacted:valid {
        border: 1px solid green;
    }
</style>
<form>
    <input type="text" required />
    <input type="text" />
    <input type="submit" />
</form>
<script>
    var inputs =
document.querySelectorAll('input[type=text]');
    for (var i = 0; i < inputs.length; i++) {
        inputs[i].addEventListener('blur',
function(event) {

event.target.classList.add('interacted');
        }, false);
    }
</script>
```

Firefox a compris cette problématique et implémenté deux pseudoclasses additionnelles (Lien 60) : :-moz-ui-invalid et :-moz-ui-valid. À la différence de :invalid et :valid, ces pseudoclasses ne cibleront pas d'éléments tant que ceux-ci n'ont pas été modifiés ou que l'utilisateur tente de valider le formulaire.
Cette évolution a été intégrée dans la spécification des sélecteurs CSS niveau 4 (Lien 61) avec le sélecteur :user-error qui fonctionne quasiment comme :-moz-ui-invalid. Il ne reste plus qu'à attendre l'implémentation par les navigateurs.
Pour terminer, faites attention que :invalid et :valid sont supposés cibler les balises <form> aussi bien que les champs. Actuellement, seul Firefox le fait, mais pensez-y lorsque vous définissez des règles globales pour ces pseudoclasses.

6.6. Navigateurs non compatibles

L'API de validation est plutôt bien supportée par les navigateurs, mais il en existe malgré tout qui ne sont pas compatibles. Parmi eux, les versions d'Internet Explorer inférieures à 8, Safari de iOS et le navigateur Android par défaut.

7. Gérer les navigateurs non compatibles

Si vous souhaitez utiliser l'API de contrainte de validation sur un formulaire d'un site en production, vous devez obligatoirement gérer les navigateurs non compatibles. D'après mon expérience, deux solutions sont possibles.

7.1. Option 1 : se baser uniquement sur la validation côté serveur

Il est important de se souvenir que même avec l'API de validation, il est nécessaire d'effectuer des vérifications côté serveur. Les utilisateurs mal intentionnés peuvent facilement contourner toutes les validations côté client et le protocole HTTP n'impose pas qu'une requête vienne d'un navigateur.
De ce fait, la validation côté client ne peut être considérée que comme une amélioration progressive pour l'utilisateur. Tout formulaire doit être utilisable même en l'absence de validation par le navigateur.
Puisque vous devez obligatoirement valider les valeurs reçues côté serveur, cette validation peut renvoyer des messages d'erreur qui seront affichés à l'utilisateur et cela peut être considéré comme une solution valable pour les navigateurs n'implémentant pas l'API de validation.

7.2. Option 2 : solutions de remplacement

Si se contenter de la validation côté serveur est suffisant pour certaines applications, pour beaucoup d'autres, se passer de validation côté client pour les navigateurs non conformes n'est pas envisageable.
Si vous souhaitez utiliser cette API et que les navigateurs non compatibles se comportent de façon identique, le meilleur moyen est d'utiliser un *polyfill* (Lien 62). En plus de permettre de simuler les fonctionnalités de l'API pour les navigateurs non conformes, la plupart de ces solutions de remplacement vont plus loin et corrigent certains défauts des implémentations natives.

8. Solutions de remplacement

Il existe de nombreux scripts de remplacement (Lien 63) vous permettant d'utiliser l'API de validation quel que soit le navigateur. Je vais vous présenter deux des plus populaires.

8.1. Webshims

Webshims (Lien 64) est une collection de scripts de remplacement dont une partie correspond aux formulaires HTML5 et à l'API de validation.
Pour montrer comment fonctionne Webshims, reprenons le premier exemple avec un simple champ texte ayant l'attribut required.

```
<form>
    <input type="text" required value="" />
    <input type="submit" value="Submit" />
</form>
```

Pour permettre de le faire fonctionner sur tous les navigateurs, nous devons inclure Webshims et les scripts

dont il dépend dans la page, puis appeler $.webshims.polyfill('forms').

```
<script src="js/jquery-1.8.2.js"></script>
<script src="js/modernizr-yepnope-
custom.js"></script>

<script src="js-
webshim/minified/polyfiller.js"></script>

<script>jQuery.webshims.polyfill('forms');</scrip
t>

<form>
    <input type="text" required value="" />
    <input type="submit" value="Submit" />
</form>
```

Pour les navigateurs compatibles, il n'y aura aucun changement dans le résultat. En revanche, les navigateurs non compatibles vont maintenant pouvoir empêcher la soumission du formulaire et afficher un message d'erreur. Voici ce que cela rendra dans certains navigateurs :

En plus de rendre compatibles tous les navigateurs, Webshims propose des solutions à beaucoup de limitations dont nous avons parlé :

- spécifier des messages d'erreur via l'attribut data-errormessage ;
- mise en œuvre de classes form-ui-valid et form-ui-invalid qui se comportent comme :-moz-ui-valid et :-moz-ui-invalid ;
- mise en œuvre d'événements firstinvalid, lastinvalid, changedvalid, et changedinvalid ;
- gestion correcte des faux positifs de Webkit lors de la soumission.

Pour plus d'informations sur les possibilités offertes par Webshims, regarder la documentation sur les formulaires HTML5 : Lien 65.

8.2. H5F

H5F est une solution de remplacement légère et ne dépendant d'aucun autre script. Elle implémente l'API de validation complète ainsi qu'un bon nombre de nouveaux attributs.

Voici comment intégrer H5F dans votre page et gérer un formulaire simple :

```
<script src="H5F.js"></script>

<form>
    <input type="text" required value="" />
    <input type="submit" value="Submit" />
</form>
<script>

H5F.setup(document.getElementsByTagName('form'));
</script>
```

H5F va empêcher la soumission du formulaire s'il contient des données non valides, mais les utilisateurs ne verront la bulle d'information que pour les navigateurs qui supportent nativement cette fonctionnalité. Comme H5F reproduit l'API de validation complète, vous pouvez gérer l'interface utilisateur comme bon vous semble.
Vous pouvez consulter la page d'exemples de H5F (Lien 66) pour voir comment ajouter une bulle d'information à droite des champs concernés. J'ai aussi créé un exemple montrant comment afficher une liste de messages d'erreur (Lien 67) en haut du formulaire.
En plus d'intégrer une solution de remplacement pour l'API de validation, H5F propose des classes reproduisant le comportement de :-moz-ui-valid et :-moz-ui-invalid. Par défaut, ces classes sont valid et error, mais vous pouvez les personnaliser dans le second paramètre de H5F.setup.

```
H5F.setup(document.getElementById('foo'), {
    validClass: 'valid',
    invalidClass: 'invalid'
});
```

Pour plus d'informations sur H5F, consulter la page Github du projet : Lien 68.

9. Conclusion

L'API HTML5 de contrainte de validation permet d'ajouter une validation des champs de formulaires côté client tout en mettant à disposition une API JavaScript et des règles CSS pour la personnalisation.
Bien qu'il existe quelques problèmes concernant l'implémentation et la gestion des navigateurs anciens, avec une bonne solution de remplacement ou une prise en charge alternative côté serveur, vous pouvez d'ores et déjà utiliser cette API dans vos formulaires.

Retrouvez l'article de TJ VanToll traduit par Didier Mouronval en ligne : Lien 69

Envoyer des chaînes ou des structures par PostMessage

Ce tutoriel a pour but d'expliquer une façon simple d'envoyer des chaînes de caractères et des structures complexes en asynchrone par PostMessage.

1. Introduction

Il est fréquent de vouloir notifier à l'utilisateur l'avancement d'une tâche en lui présentant un certain nombre d'informations à l'écran. Malheureusement, la plupart des méthodes proposées sont synchrones, que ce soit par *Synchronize*, WM_COPYDATA, fichier mappé et par conséquent entraîne un arrêt du *thread* secondaire le temps du traitement par le *thread* principal.
Ce tutoriel va vous expliquer comment mettre en œuvre une façon simple d'envoyer n'importe quel type de données en asynchrone par *PostMessage* à l'aide de la table d'atomes.

2. La table d'atomes

Qu'est-ce qu'un atome ? Un atome est un identificateur sur 16 bits d'une chaîne de caractères. La table d'atomes est donc simplement une liste de chaînes.
La chaîne peut être « numérique » si le texte est de la forme #1234. L'atome représente alors cette valeur. Sinon elle est de type « texte ». Les atomes « textes » et « numériques » ont chacun leur zone propre dans la table atomique. Les « numériques » vont de $0001 à $BFFF (*MAXINTATOM* -1) alors que les « textes » de $C000 (*MAXINTATOM*) à $FFFF.
Chaque chaîne est unique. Enregistrer plusieurs fois la même chaîne, indépendamment de la casse, renverra toujours le même atome. La table d'atomes nous permet ainsi de partager une même chaîne par l'intermédiaire de son identificateur.
Une chaîne est limitée à 255 octets. Soit 255 caractères en ANSI ou 127 en Unicode.
À l'instar de la gestion des chaînes sous Delphi, un compteur de référence est utilisé. Chaque ajout doit donc obligatoirement être suivi d'une libération. L'atome n'est effectivement supprimé que lorsque le compteur est à 0. Ce compteur ne s'applique qu'aux chaînes « textes ». Les « numériques » n'en possèdent pas.
Les tables sont gérées par le système. Il y en a une globale au système et une par processus. La taille d'une table est de 64 ko (atome max. = $FFFF).
Même sans le savoir, nous utilisons régulièrement la table atomique pour :

- stocker les formats du presse-papier ;
- la communication *DDE* ;
- les messages enregistrés par *RegisterWindowMessage* ;
- etc.

3. Ajouter, lire et libérer un atome

Il y a deux groupes de fonctions : un pour les atomes locaux (propres au processus) et son pendant pour les globaux : *AddAtom* (*GlobalAddAtom*), *DeleteAtom* (*GlobalDeleteAtom*), *GetAtomName* (*GlobalGetAtomName*), etc.
Nous ne nous intéresserons ici qu'à la table globale. Nous pourrons ainsi aussi envoyer des informations entre processus.
Le principe est très simple : la source ajoute une chaîne à la table et envoie l'atome à la cible. La cible récupère la chaîne associée à l'atome et la supprime de la table.

3.1. Ajouter et envoyer une chaîne courte

Ce que nous appelons ici « chaîne courte » ne correspond pas exactement à la définition de Delphi. Elle s'exprime en octets et non en caractères avec une limite de 255 octets. Elle correspond donc à une chaîne de maximum 255 caractères en ANSI et 127 caractères en Unicode.

Dans une communication interprocessus, il faudra bien sûr prendre soin de créer un message par *RegisterWindowMessage*. Ici, nous utiliserons simplement WM_USER.

Ajouter et envoyer

```
procedure PostText(aWnd :hWnd; aText :string);
var
  Atom :TAtom;

begin
  Atom := GlobalAddAtom(PChar(aText));

  if not PostMessage(aWnd, WM_USER, Atom, 0) then
     GlobalDeleteAtom(Atom);
end;
```

3.2. Lire et libérer

À la réception du message, l'application cible va récupérer la chaîne associée à l'atome et supprimer le texte de la table atomique.

La table n'ayant pas une capacité illimitée, il est très important de ne pas oublier la suppression.
De plus et puisque nous utilisons la table globale, nos entrées ne seraient pas supprimées à l'arrêt du processus et la seule solution pour l'utilisateur dans le cas d'une table pleine serait de fermer sa session !

Lire et libérer

```
type
  TForm1 = class(TForm)
  protected
    procedure WMGetTextMessage(var Message
```

```
:TMessage); message WM_USER;
  end;

procedure TForm1.WMGetTextMessage(var Message:
TMessage);
var
  Len  :byte;
  Atom :TAtom;
  Text :string;

begin
  //Chaîne de max 255 octets
  SetLength(Text, MAXBYTE div SizeOf(Char));

  //Lecture de la table et récupération de la
longueur réelle
  Len := GlobalGetAtomName(Message.WParam,
PChar(Text), Length(Text));

  //Longeur réelle
  SetLength(Text, Len);

  //Supression de la chaîne
  GlobalDeleteAtom(Message.WParam);

  //Traitement
end;
```

4. Envoyer une chaîne longue ou une structure

Nous avons vu jusqu'ici comment envoyer une chaîne courte. Voyons maintenant comment envoyer une chaîne de plus de 255 octets ou carrément une structure plus complexe comme un *record* ou un *stream*.
Une chaîne de caractères n'étant finalement rien d'autre qu'une suite d'octets, nous allons simplement mettre en place un système de sérialisation des données en découpant notre variable en bloc de *N* bytes et en y ajoutant un atome représentant le bloc suivant. Les fonctions attendant des *PChar* en paramètre, nous utiliserons des pointeurs non typés (*@Data*) pour passer au travers du compilateur.
Et puisque nous ne nous intéressons qu'à des octets, nous utiliserons les fonctions ANSI : *GlobalAddAtomA*, *GlobalDeleteAtomA*, etc.
La structure dans le cas d'une chaîne longue se présenterait donc ainsi :

```
const
  MaxChars = (MAXBYTE -SizeOf(TAtom)) div
SizeOf(Char);

Type
  TData = packed record
    Next  :TAtom;
    Chars :array[0..MaxChars -1] of char;
  end;
```

Mais nous allons directement pousser plus loin le concept et créer une structure plus complète qui acceptera n'importe quel type de donnée.

```
const
  MaxBytes = (MAXBYTE -SizeOf(TAtom) -2) div 2;

type
  TData = packed record
    Next  :TAtom;
    Len   :byte;
    Bytes :string[MaxBytes *2];
    Null  :ansichar;
  end;
```

Le *record* contient un atome sur le bloc suivant (*Next*), la taille du bloc (*Len*), le bloc de données (*Bytes*) et un caractère de fin de chaîne #0 (*Null*).

Définir la longueur d'une chaîne dans la déclaration d'une variable (*string[]*) la transforme automatiquement en tableau de *AnsiChar*, même si l'application est *Unicode* !

Il ne faut pas oublier que les atomes représentent des chaînes à zéro terminal. À part la variable *Null* qui assure la validité de la pseudochaîne, aucun octet du *record* ne doit être à zéro sous peine de se retrouver avec une donnée tronquée, voire une violation d'accès !

Puisque notre pseudochaîne n'accepte pas les zéros, nous allons ajouter deux fonctions pour convertir nos octets en caractères « 0 » à « F » et inversement. Un octet de donnée réelle utilisera donc deux octets à l'envoi (d'où le *div 2* pour le calcul de *MaxBytes*).

Conversion caractères ↔ octet

```
const
  Codes : array[0..$F] of ansichar =
'0123456789ABCDEF';

function StrToByte(aText :string; aIndex
:integer) :byte; inline;
begin
  Result := ((Pos(aText[aIndex], Codes) -1) shl
4) or
            (Pos(aText[aIndex+1], Codes) -1);
end;

function ByteToStr(aByte :byte) :string; inline;
begin
  Result := Codes[aByte shr 4]
           +Codes[aByte and $F];
end;
```

Pour la même raison, *Next* (l'atome suivant) ne peut pas être à zéro pour signifier la fin de la donnée. Nous ne pouvons pas non plus prendre n'importe quelle autre valeur pour ne pas accidentellement lire un atome sans rapport. La table utilisable allant de *MAXINTATOM* ($C000) à $FFFF, nous allons choisir *MAXINTATOM* -1 ($BFFF). Nous appellerons cette constante *EOD* (*End Of Data*).

End Of Data

```
const
  EOD = MAXINTATOM -1;  //End Of Data ($BFFF)
```

Par sécurité, nous déclarerons encore une table contenant tous les atomes créés. En cas de problème dans l'application, elle nous permettra de nettoyer la table à la fermeture et ainsi de ne pas obliger l'utilisateur à quitter sa session.

Chaque élément représente un compteur de référence de l'atome concerné. Il ne faut pas oublier que deux chaînes identiques renvoient le même atome et par conséquent,

plusieurs de nos données en attente de traitement pourraient partager le même !

Table atomique temporaire

```
var
  GlobalAtoms :array[MAXINTATOM..$FFFF] of byte;

procedure ClearPostMessageTable;
var
  i :integer;

begin
  for i := Low(GlobalAtoms) to High(GlobalAtoms)
do
    while GlobalAtoms[i] > 0 do
    begin
      GlobalDeleteAtom(i);
      Dec(GlobalAtoms[i]);
    end;
end;

initialization
  ZeroMemory(@GlobalAtoms, SizeOf(GlobalAtoms));

finalization
  if ClearTableOnExit then
    ClearPostMessageTable;
```

À la finalisation, *ClearPostMessageTable* est conditionné par une variable booléenne *ClearTableOnExit* au cas où les messages devaient persister après la sortie du programme. Par exemple pour une application console qui notifie un résultat et quitte immédiatement. *ClearTableOnExit* est fixé à vrai par défaut.

ClearTableOnExit

```
var
  ClearTableOnExit :boolean = TRUE;
```

5. Cas général

5.1. Envoyer un buffer

Cette fonction est la base du système de découpe et d'envoi de donnée. Par simplification, la donnée est traitée depuis la fin. La traiter depuis le début nous obligerait à passer par un tableau temporaire et de remplir les *Next* dans une deuxième passe.
Au cas où une erreur surviendrait pendant le traitement (la cause la plus probable étant un dépassement de capacité de la table atomique), une liste d'atomes locale est utilisée et permet la libération immédiate des atomes déjà créés.
La fonction renvoie 0 (ERROR_SUCCESS) si elle s'est bien déroulée, sinon le code d'erreur.

Envoi d'un buffer

```
function PostBufferMessage(aWnd :hWnd;
aMessage :cardinal; aBuffer :PByte; aLen
:integer) :integer;
var
  Data   :TData;
  Atoms :array of TAtom;
  Count :integer;
  i      :integer;

begin
  ZeroMemory(@Data, SizeOf(Data));

  //La donnée est traitée depuis la fin => Next =
EOD
  Data.Next := EOD;

  try
    //S'il n'y a pas de donnée, envoie une chaîne
vide
    if Assigned(aBuffer) then
    begin
      //Table d'atomes nécessaires à l'envoi de
la donnée.
      //Si une erreur survient en cours de
traitement, permet
      //de libérer les atomes déjà créés.
      SetLength(Atoms, aLen div MaxBytes +1);
      Count := 0;

      //Traîte la donnée depuis la fin. Le faire
depuis le début
      //nous obligerait à utiliser une table
temporaire et de
      //renuméroter les <Next> dans une deuxième
passe.
      for i := aLen -1 downto 0 do
      begin
        //Conversion byte -> caractères
        Data.Bytes := ByteToStr(aBuffer[i])
+Data.Bytes;

        //Taille max atteinte => Stocke la chaîne
        if i mod MaxBytes = 0 then
        begin
          Data.Len  := Length(Data.Bytes) div 2;
          Data.Next := GlobalAddAtomA(@Data);

          //Si erreur, Next = 0. Sinon ajoute
l'atome à nos listes
          if Data.Next <> 0 then
          begin
            Atoms[Count] := Data.Next;
            inc(GlobalAtoms[Data.Next]);
            inc(Count);

            //Reset pour prochaine boucle
            Data.Bytes := '';
          end
          else Exit(GetLastError);
        end;
      end;
    end;

    //Envoi
    if PostMessage(aWnd, aMessage, Data.Next,
aLen)
    then Result := ERROR_SUCCESS
    else Result := GetLastError;

  finally
    //Libération des atomes déjà créés si erreur
    if Result <> ERROR_SUCCESS then
      for i := 0 to Count -1 do
      begin
        GlobalDeleteAtom(Atoms[i]);
        dec(GlobalAtoms[Atoms[i]]);
      end;
  end;
end;
```

Pour envoyer un *record*, nous invoquerons cette fonction

ainsi :

Envoi d'un record

```
type
  TRec = record
    Val1  :word;
    Val2  :dword;
    Val3  :extended;
    Val4  :array[0..100] of char;
  end;

var
  Rec :TRec;

procedure TForm1.Button1Click(Sender: TObject);
var
  Wnd   :hWnd;
  Error :integer;
begin
  Wnd := FindWindow('DestClass', 'DestName');

  if Wnd <> 0 then
  begin
    Error := PostBufferMessage(Wnd, WM_USER,
@Rec, SizeOf(Rec));

    if Error <> ERROR_SUCCESS then
      Raise
Exception.Create(SysErrorMessage(Error));
  end;
end;
```

5.2. Remplir un buffer

Maintenant que nous avons envoyé une donnée, nous allons voir comment la récupérer et la décoder.
La fonction renvoie la taille de la donnée. Si le buffer n'est pas spécifié, elle nous permet d'allouer un buffer avant un deuxième appel.

Remplir un buffer

```
function GetBufferMessage(aAtom :TAtom;
aBuffer :PByte; aLen :integer) :integer;
var
  Data :TData;
  i    :integer;

begin
  //Result renvoie la taille de la donnée
  Result := 0;

  //Lit tant que "End Of Data" n'est pas atteint
  while aAtom <> EOD do
    if GlobalGetAtomNameA(aAtom, @Data,
SizeOf(Data)) <> 0 then
    begin
      inc(Result, Data.Len);

      //Si le buffer n'est pas spécifié (nil),
l'atome n'est pas supprimé
      //et la fonction sert uniquement à
récupérer la taille totale de la
      //donnée en vue de l'allocation d'un buffer
      if Assigned(aBuffer) then
      begin
        //Supprime l'atome
        GlobalDeleteAtom(aAtom);
        dec(GlobalAtoms[aAtom]);
        i := 1;

        if Data.Len < aLen then
          aLen := Data.Len;

        //Convertit la chaîne en octets
        while i < aLen *2 do
        begin
          aBuffer^ := StrToByte(Data.Bytes, i);
          inc(i, 2);
          inc(aBuffer);
        end;
      end;

      //Atome suivant
      aAtom := Data.Next;
    end
    else
    begin
      Result := 0;
      Break;
    end;
end;
```

Et pour récupérer notre *record*.

Récupérer un record

```
  TForm1 = class(TForm)
  protected
    procedure WMGetBuffertMessage(var Message
:TMessage); message WM_USER;
  end;

procedure TForm1.WMGetBuffertMessage(var Message:
Tmessage);
var
  Rec :TRec;

begin
  GetBufferMessage(Message.WParam, @Rec,
SizeOf(Rec));

  //Traitement
end;
```

6. Chaînes longues et streams

Maintenant que les fonctions génériques par buffer sont créées, il est facile de les encapsuler pour d'autres types de données.

6.1. Chaînes longues

Envoi.

Envoi d'une chaîne longue

```
function PostTextMessage(aWnd :hWnd; aMessage
:Cardinal; aText :string) :integer;
begin
  Result := PostBufferMessage(aWnd, aMessage,
@aText[1], Length(aText) *SizeOf(Char));
end;
```

Réception.

Réception d'une chaîne longue

```
function GetTextMessage(aAtom :TAtom) :string;
var
  Len :integer;
```

```
begin
  Len := GetBufferMessage(aAtom, nil, 0);
  SetLength(Result, Len div SizeOf(Char));
  GetBufferMessage(aAtom, @Result[1], Len);
end;
```

6.2. Streams

Envoi.

Envoi d'un stream

```
function PostStreamMessage(aWnd :hWnd;
aMessage :Cardinal; aStream :TStream) :integer;
var
  Buffer :array of byte;

begin
  SetLength(Buffer, aStream.Size);
  aStream.Position := 0;
  aStream.Read(Buffer[0], aStream.Size);

  Result := PostBufferMessage(aWnd, aMessage,
@Buffer[0], aStream.Size);
end;
```

Réception.

Réception d'un stream

```
function GetStreamMessage(aAtom :TAtom;
aStream :TStream) :integer;
var
  Buffer :array of byte;

begin
  Result := GetBufferMessage(aAtom, nil, 0);
  SetLength(Buffer, Result);
  GetBufferMessage(aAtom, @Buffer[0], Result);

  aStream.Write(Buffer[0], Result);
end;
```

7. Limitations

Il n'y a en fait qu'une seule réelle limitation : le nombre d'atomes disponibles dans la table atomique.
S'il n'y avait aucun atome utilisé, la taille maximale de la donnée serait de $FFFF -$C000 (*MAXINTATOM*) *125 (*MaxBytes*) +1, soit 2 048 000 octets. Dans la réalité, ce ne sera évidemment pas le cas. D'autres applications en auront déjà consommé quelques-uns et il n'y a aucun moyen de savoir combien !
Une table atomique pleine renverra l'erreur 8 (*ERROR_NOT_ENOUGH_MEMORY*) : *Espace insuffisant pour traiter cette commande*.
À prendre aussi en considération :

- le *broadcasting* (*HWND_BROADCAST*) n'est pas supporté par cette méthode puisque le premier « lecteur » va effacer la donnée ;
- le principe du codage/décodage est assez lent (pourrait être optimisé) et même s'il est théoriquement possible d'envoyer jusqu'à 2 MB, je ne conseille pas de dépasser les quelques centaines de kilos. Asynchrone oui, mais faut pas pousser !

8. Conclusion

Ce tutoriel est maintenant terminé !
Je pense que nombre d'entre vous auront découvert les atomes et certainement cette façon détournée de les utiliser.
L'unité *MessageEx.pas* est disponible au format ZIP. Il vous suffit de l'ajouter à la clause *uses* pour l'utiliser dans votre application et de laisser aller votre imagination pour envoyer du texte, des images et tout ce qui vous passera par la tête !
Sources : Lien 70

Retrouvez l'article d'Andnotor en ligne : Lien 71

Accélérer le téléchargement de fichiers

Tutoriel sur l'accélération du téléchargement de fichiers à l'aide de plusieurs connexions et d'un fichier mappé.

1. Introduction

Qui n'a jamais trouvé le téléchargement d'un fichier de plusieurs dizaines de mégas extrêmement long malgré une connexion Internet haut débit ? Vous ? Moi ?… Tout le monde !
Ce tutoriel va vous expliquer comment en accélérer le téléchargement à l'aide de plusieurs connexions simultanées en *multi-threads* et d'un fichier mappé.
Remarque :
Il ne s'agit pas de transformer un accès Internet à 15 Mb/s en un à 25 (faut pas rêver) mais simplement de passer outre la limitation de bande passante définie par l'hébergeur du site visé. Il faut par conséquent que le débit descendant de votre connexion soit supérieur à la limite en place. Dans le cas contraire, vous ne remarquerez aucune différence !

2. Introduction aux fichiers mappés

Ce tutoriel n'a pas pour but d'expliquer tous les détails du mappage de fichier. En voici seulement une courte introduction.
Un fichier mappé est l'association entre le contenu d'un fichier et une zone de la mémoire virtuelle d'un ou de plusieurs processus. C'est un pont entre le fichier et les processus.
Une *vue* est la portion de la zone de mémoire virtuelle qu'un processus utilise pour accéder au contenu du fichier. Elle peut représenter le fichier complet ou une partie seulement. En d'autres termes, une *vue* est un pointeur sur un fragment du fichier chargé dans la mémoire du processus.

3. Principe

Sans grande surprise, le but va être de lancer plusieurs connexions Internet simultanément et à travers chacune d'elles de télécharger un fragment du fichier.
Chaque connexion est réalisée dans un *thread* séparé (*TViewDownloadThread*). Ils sont créés au démarrage et

synchronisés par *Events*.
Un *thread* principal (*TDownloadThread*) se charge de répartir le travail en fonction de l'occupation des *threads* de téléchargement.
Le téléchargement est réalisé à l'aide du composant INDY : *TIdHttp*.

4. Nombre de tâches

Le nombre de tâches peut être supérieur au nombre de processeurs de la machine. On peut en effet partir du principe que le dialogue client/serveur à travers Internet est lent et que chaque *thread* passera un certain temps à attendre des réponses.
Il y a cependant certaines limitations qu'il serait inutile de dépasser :

- une vue ne peut pas commencer à n'importe quel endroit d'un fichier mappé, elle doit respecter un alignement déterminé par le système. Cet alignement est récupéré à l'aide de l'API *GetNativeSystemInfo*. Il correspond à la « granularité d'allocation » et est normalement de 64 kB sous Windows.
  Dès lors, le nombre de *threads* ne devrait excéder : *la taille du fichier à télécharger div 64 kB + 1*. Si le fichier faisait 200 kB, le nombre de *threads* utilisés serait de quatre, même si on en créait dix.
  Dans notre exemple, la taille d'un bloc sera fixée à 15 fois l'allocation minimale. Un *thread* téléchargera donc environ 975 kB ce qui est un bon compromis entre l'alignement obligatoire des vues et le temps de latence. Il faudra cependant réduire ce multiplicateur si le débit de la connexion Internet est inférieur à 15 Mb/s.

  Voici comment récupérer la taille d'allocation minimale :

Taille d'allocation minimale
```
var
  SystemInfo :TSystemInfo;
  AllocSize  :Int64;

begin
  GetNativeSystemInfo(SystemInfo);
  AllocSize :=
SystemInfo.dwAllocationGranularity;
end;
```

- la vitesse de la connexion Internet (*downstream*). 10 Mb/s permettraient dans le meilleur des cas une quinzaine de *threads* si 64 kB était utilisé ;
- la qualité du service fourni par l'hébergeur du site. Limite de bande passante et du nombre de connexions concurrentes, temps de latence, etc. Certains serveurs n'acceptent tout simplement pas le téléchargement par fragment !

Créer plus de tâches que nécessaire a cependant un impact négligeable sur la fonction proposée ci-dessous. Les inutilisées sont placées en mode attente infinie par *WaitForSingleObject(INFINITE)* et ne consomment par conséquent aucun temps processeur.

5. Tâche de téléchargement

Les tâches de téléchargement sont en charge de récupérer un fragment du fichier. Elles pourraient être créées uniquement lorsqu'un nouveau bloc est requis. Ici, elles sont créées une fois pour toutes au démarrage et synchronisées par *Events*.

Déclaration
```
Type
  TViewDownloadThread = class(TThread)
  private
    URL          :string;
    MapFile      :THandle;
    View         :TViewInfo;

    Stream       :TMemoryStream;
    Http         :TIdHttp;
    RunEvent     :THandle;
    ReadyEvent   :Thandle;
  end;
```

- *URL* est le fichier à télécharger.
- *MapFile* est le fichier mappé auquel nous appliquerons une vue.
- *View* est une structure contenant l'offset et la taille du bloc à télécharger.
- *Stream* est le buffer de téléchargement.
- *Http* est un *TidHttp* en charge de la connexion au serveur.
- *RunEvent* est un *Event* local ordonnant le démarrage de la fonction en attente.
- *ReadyEvent* est un *Event* fourni par le *thread* principal et signifie la fin du téléchargement du bloc et l'état prêt pour une nouvelle commande.

TViewInfo est déclaré ainsi :

TViewInfo
```
  TViewInfo = record
    Size :int64;

    case byte of
      0 : (Offset :int64);
      1 : (LoOffset, HiOffset :integer);
  end;
```

Size est la taille du bloc à télécharger. Le *case* permet de lire/écrire l'offset sous forme d'un *int64* ou de deux entiers *LowInteger* et *HighInteger* sans devoir appliquer de transformation. *MapViewOfFile* que nous utiliserons bientôt requiert la deuxième forme.

Retrouvez la suite de l'article d'Andnotor en ligne : Lien 72

Une seconde mise à jour, Qt Creator 2.6.2 - Corrections des blocages de l'éditeur

La version 2.6 apportant les kits (ensemble de chaînes de compilation définissant la bibliothèque Qt, le compilateur et le débogueur), le support des projets Android et BlanckBerry et de nombreuses améliorations en tout genre reçoit sa seconde mise à jour à l'occasion de la sortie de Qt 5.0.1. : Lien 73

Cette version corrige les blocages que l'on pouvait rencontrer dans des cas précis d'édition de code C++.

Comme toujours, vous pouvez lire le changelog complet : Lien 74.

La nouvelle version est déjà disponible dans les binaires de Qt 5.0.1 ou comme logiciel indépendant sur la page de téléchargements de Qt : Lien 75.

Avez-vous déjà rencontré un blocage de l'éditeur ?

Commentez cette news d'Alexandre Laurent en ligne : Lien 76

Sortie de Qt 5.0

À proximité de Noël mais aussi d'une nouvelle fin du monde, Digia dépose un gros cadeau sous le sapin : Qt 5. Pour rappel, Qt est le standard *de facto* pour les interfaces graphiques en C++, mais il fournit également un bon nombre d'autres fonctionnalités (comme le support de WebKit, pour afficher des pages Web, des fonctionnalités multimédia, de réseau, de script, des interfaces déclaratives avec Qt Quick, etc., sans oublier l'EDI Qt Creator). Pour ceux qui en suivent l'actualité, il est également important de noter qu'il s'agit de la première version majeure depuis que Trolltech, la société l'ayant créé, a été rachetée, d'abord par Nokia, puis par Digia.

Voir la vidéo : Lien 77

Fonctionnalités majeures

Cette version est disponible comme un paquet de binaires pour toutes les plateformes desktop (Linux 32 et 64 bits, Mac OS X 10.7 et 10.8, Windows), comprenant, en plus de Qt 5, l'environnement de développement Qt Creator 2.6, les exemples et la documentation complète.

Les fonctionnalités majeures sont rassemblées dans une vidéo, entièrement générée par une application Qt 5, avec Qt Quick, OpenGL et WebKit. (Les sources sont bien évidemment disponibles : Lien 78.)

Voir la vidéo : Lien 79

Également à noter : la deuxième beta a apporté une totale refonte de la documentation, elle est maintenant plus adaptée à la modularisation du framework.

Compatibilité

Qt 5 est un grand remaniement de Qt 4 (sorti en 2005), tout en restant dans la continuité : le but est d'avoir un framework ouvert sur le futur, prêt à supporter toutes les plateformes, mais sans rupture forte avec Qt 4, afin de faciliter la migration.

Cette version a donc été l'occasion de bien nettoyer l'architecture interne, notamment avec une modularisation du framework, mais une migration très rapide de la plupart des applications est possible. Ceci implique notamment que les widgets – la base des interfaces graphiques des précédentes versions – sont toujours supportés, bien qu'ils ne soient plus inclus dans le module Qt GUI (ils ont été déplacés dans un autre module, afin d'en faciliter la maintenance (ils ne sont plus la seule et l'unique manière de créer des interfaces).

Un exemple de cette très forte compatibilité : Qt Creator. Une seule base de code est utilisée pour compiler tant avec Qt 5 que Qt 4.

Futur

Pour la série 5.0, encore quelques points seront régularisés : pas de paquet de binaires disponible pour MinGW (le port de GCC pour Windows), étant donné que WebKit ne le supporte pas complètement, ni pour Visual C++ 2012 (il faut compiler Qt 5 soi-même). Il est actuellement prévu de résoudre ces problèmes pour Qt 5.0.1, prévu fin janvier.

Au-delà cette première échéance, la version 5.1 est prévue pour le printemps 2013, où le but sera de stabiliser encore plus le framework et de porter des modules add-ons tels que Qt 3D et Qt Sensors dans les modules principaux. Il est également prévu de montrer les avancées des ports vers Android et iOS.

Pour les versions suivantes, il est prévu que deux versions mineures sortent chaque année.

Contribution

Bien évidemment, il s'agit d'une première version finale, expurgée des bogues les plus problématiques, mais toute une série de problèmes est connue : Lien 80. Si vous souhaitez contribuer au framework (en rapportant de nouveaux bogues, en en corrigeant, en développant de nouvelles fonctionnalités, etc.), n'hésitez surtout pas : Lien 81.

Déjà un grand nombre de gens ont contribué à cette

version : 427 personnes pour le code source. Bien plus pour le rapport de bogues, les discussions, le feedback. Avec un peu plus d'un an, le Qt Project s'est montré être une grande communauté, une grande réussite pour l'avenir du framework.

Télécharger Qt 5 : Lien 82

Commentez la news de Thibaut Cuvelier en ligne : Lien 83

Commentez également la news d'Alexandre Laurent sur la sortie de Qt 5.0.1 en ligne : Lien 84

Les derniers tutoriels et articles

Portage de Qt 4 vers Qt 5

Le portage d'applications de Qt 4 vers Qt 5 est intentionnellement simple. Il y a eu un effort consciencieux au cours du développement de Qt 5 jusqu'ici pour maintenir la compatibilité comparée à Qt 4.

1. L'article original

Le blog KDAB est rédigé par les ingénieurs de KDAB s'occupant des formations, de la consultance ainsi que du développement (de Qt et de produits additionnels). Vous pouvez trouver les versions originales : Lien 85.

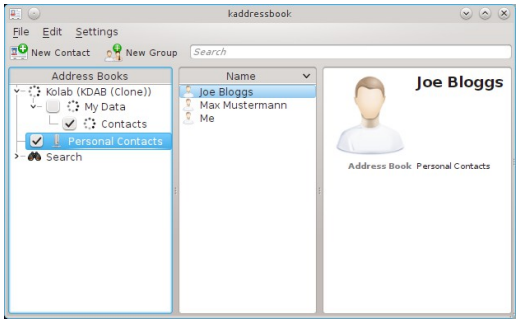
Cet article est une traduction de l'article original écrit par Stephen Kelly paru le 11 juin 2012 : Lien 86.

Cet article est une traduction de l'un des articles en anglais écrits par KDAB. Les éventuels problèmes résultant d'une mauvaise traduction ne sont pas imputables à KDAB.

2. Introduction

Par rapport au portage de Qt 3 vers Qt 4, les classes centrales n'ont pas connu de gros nettoyages de leurs API ni de nouvelles bibliothèques en remplacement des anciennes (comme QPtrList et QValueList remplacées par QList, les « itemwiews » remplacent Q3ListView et les « graphicsviews » remplacent les API Canvas) et les modifications qui compilent, mais affectent le comportement à l'exécution, comme la méthode QWidget ::show qui devient non virtuelle et la méthode « painting » qui devient restrictive sur le paintEvent.

Certes, l'effort de portage n'est pas nul, cet article résume quelques-unes des étapes requises pour porter KDE vers Qt 5.



KDE PIM est une des dernières parties à avoir été portée complètement vers Qt 4 et kdelibs 4. Le portage vers Qt 5 sera donc plus rapide.

3. Avant le portage

Dans la stratégie de portage, il convient d'effectuer une compilation du code avec l'ancienne et la nouvelle version de Qt. Ceci permettra alors de contrôler les parties du code qui devront être modifiées lors du portage. Il faudra également s'assurer que les tests unitaires continuent à fonctionner durant le temps de portage, les régressions résultant du portage du code seront facilement distinguées des bogues introduits par Qt 5.

3.1. Portage du support de Qt 3

Le portage des logiciels vers Qt 5 peut démarrer par la modernisation de la base du code vers Qt 4.

Un des changements les plus marquants de Qt 5 d'un point de vue portage d'une ancienne base de code est la suppression du module Qt3Support et la suppression des API de ce module. Dans la plupart des cas, le code de Qt3Support consiste en une méthode qui a été renommée dans Qt 4. L'ancienne méthode appelle la nouvelle, comme la méthode QWidget::setShow() qui appelle la méthode QWidget::setVisible(). Certaines parties de KDE utilisent encore ces vieilles méthodes et ces mêmes parties utilisent encore des vieilles bases de code.

Le portage des API Qt3Support vers Qt 4 est une des étapes nécessaires et inévitables d'un portage de Qt 4 vers Qt 5, même s'il est possible en théorie de compiler Qt3Support avec Qt 5 pour certains points.

3.2. Correction des fichiers d'en-têtes

Un des changements majeurs dans l'infrastructure interne de Qt 5, par rapport à Qt 4, est la séparation des widgets du module Qt Gui vers un nouveau module, Qt Widgets. Ceci requiert des modifications du système de compilation, mais aussi dans les inclusions de fichiers d'en-têtes qui n'étaient pas présentes auparavant ; certaines inclusions ont été supprimées des fichiers d'en-têtes qui restent maintenant dans le module Qt Gui. Une des manifestations les plus courantes est de devoir ajouter #include « QDrag » alors qu'il n'y en avait pas besoin avant. Ceci est dû à la non-inclusion du fichier d'en-tête « gui/kernel/qevent.h », mais ce n'est plus le cas dans Qt 5.

D'autres problèmes venant des inclusions dans le portage de Qt 4 vers Qt 5 sont de modifier les en-têtes des classes qui ont été déplacés vers le module Qt Widget. Tandis que la base de code Qt 4 pourrait utiliser :

```
#include <QtGui/QWidget>
```

Son équivalent Qt 5 devra utiliser :

```
#include <QtWidgets/QWidget>
```

Ou, plus probablement (qui fonctionne avec Qt 4 et Qt 5) :

```
#include <QtWidget>
```

Un petit script peut être utilisé pour effectuer les changements. Comme la suppression de l'utilisation Qt3Support, le nettoyage des inclusions peut largement être achevé dans une base de code Qt 4 avant le portage.

3.3. Correction des définitions de plateformes

Il semble y avoir quelques modifications à apporter dans les codes de base de Qt et KDE concernant les plateformes spécifiques, utilisant les macros Q_WS_* à la place de Q_OS_*. Par exemple :

```
#ifdef Q_WS_WIN
// Appel de l'API Windows
#endif
```

à la place de :

```
#ifdef Q_OS_WIN
// Appel de l'API Windows
#endif
```

Dans Qt 5, les macros Q_WS_* ont été supprimées, le code les utilisant ne pourra plus être compilé. Quand c'est approprié (c'est-à-dire quand le code contenu dans l'enveloppe est spécifique au système d'exploitation et non spécifique au système Windows), chaque code peut et doit être porté dans les macros Q_OS_*.

3.4. Macros Q_OBJECT oubliées et nettoyage des métatypes

Qt 4 peut pardonner quand une sous-classe de QObject requiert l'utilisation de la macro Q_OBJECT, seulement ceci peut apporter des bogues lors de l'exécution (Lien 87) dans le système QMetaObject. Ceci ne change pas dans Qt 5, mais, si on tente de mettre un pointeur vers un type dérivé de QObject dans un objet QVariant en utilisant la macro Q_DECLARE_METATYPE, on peut obtenir une erreur de compilation dans Qt 5 (comme dans Qt 4, mais ce sera une autre erreur de compilation). Ceci est dû à Qvariant, qui enregistre maintenant le fait que le type qui est enregistré est un pointeur vers un type dérivé de QObject, ce qui est utilisé pour des fonctionnalités intéressantes dans Qt Declarative, les langages de bindings et d'autres bibliothèques qui utilisent énormément d'API d'introspection de QmetaObject (Lien 88).

Un autre type d'effet est que l'argument typé dans la macro Q_DECDLARE_METATYPE doit être défini complètement, mais non déclaré. Comme l'exemple ci-dessous, mais qui ne compile pas :

```
class MyType;
Q_DECLARE_METATYPE(MyType);
```

La macro doit être déplacée à un emplacement où MyType est déclaré complètement (comme le fichier d'en-tête où il est déclaré). Dans certains cas où MyType est dérivé de QObject, la macro doit être retirée entièrement.

3.5. Refactoring

L'un des changements majeurs de Qt 5 est un bon focus sur QML, un langage interprété pour la création d'interfaces graphiques, et Qt Quick, l'API qui accompagne ce langage. Bien que Qt Widgets soit toujours disponible, testé et fonctionnel sous Qt 5, l'attention sur les performances et les bénéfices des interactions avec l'utilisateur ont été portés vers QML.

Comme QML est un langage interprété et ne possède pas les mêmes contraintes sur les types sécurisés dont dispose C++, il est largement destiné à être utilisé avec les modèles de données représentés par des sous-classes QObject et leurs propriétés et autres types qui peuvent être contenus dans un objet QVariant.

Si la part des buts de portage d'une base de code vers Qt 5 est d'augmenter l'utilisation de QML, alors l'accent peut être mis sur la refactorisation du code existant pour qu'il soit plus logique et plus représentatif du modèle - la représentation de l'état de l'application et de son contenu - et séparé des widgets utilisés par l'application. Ce type de refactoring peut être effectué avec un code de base Qt 4. Le travail d'un portage expérimental vers QML peut aussi être fait avec une base de code Qt 4 pour vérifier le concept de refactoring. Ceci est un bel effet de QML devenu disponible durant les versions de Qt 4 - en effet, il est présent dans la bibliothèque Qt 5 Support.

3.6. Portage depuis QWS

Le système QWS ne fait pas partie de Qt 5 et ces API (Lien 89) peuvent être supprimées. Le code utilisant ces API aura besoin d'être porté vers le nouveau système QPA, qui est la partie centrale de Qt 5. QPA est actuellement introduit dans Qt 4.8 (quoique l'API Qt 5 soit quelque peu différente).

Il est possible de porter ce code vers QPA avec Qt 4.8, avec des modifications mineures pour Qt 5, mais on aura largement le même design en haut niveau. Il n'y a pas de documentation des API de QPA dans Qt 4.8 (Lien 90), mais il existe quelques références d'implémentation pour comparaison.

4. Portage

Les étapes de portage sont décrites pour des sources compatibles avec Qt 4, ce qui veut dire que la maintenance régulière du projet de base en Qt 4 doit être terminée et qu'il est envisagé de se tourner vers Qt 5. Certaines API ont vu leurs sources modifiées avec le passage vers Qt 5, la plupart étant indiquées dans les fichiers de modifications : Lien 91. Dans la plupart des cas, ces problèmes ne sont pas pertinents sur du code « normal », car c'est rarement utilisé ou les modifications peuvent apporter des problèmes sur des cas mineurs.

Néanmoins, ces changements doivent faire partie de la stratégie de portage. Comme cela cause nécessairement des différences dans la base de code entre Qt 4 et Qt 5, il faudra alors supprimer le support de la compilation depuis Qt 4 ou entourer le code des directives de préprocesseur

#ifdef pour les autres versions.

4.1. Chargement des plug-ins

Un autre fardeau significatif du portage est le système de plug-ins lorsque le code utilisateur requiert le chargement d'un plug-in. L'outil moc est maintenant responsable de la génération des métadonnées du plug-in, via une macro du préprocesseur dans un fichier C++ (Q_EXPORT_PLUGIN2 dans Qt 4), mais dans Qt 5 une nouvelle macro doit être utilisée dans le fichier d'en-tête, que moc pourra voir.

Ce processus est décrit par Lars (Lien 92) et est relativement direct. Il faut alors modifier le code contenant la directive comme KDE avec K_EXPORT_PLUGIN : Lien 93.

Une telle enveloppe de la nouvelle macro ne serait pas possible avec Qt 5 car moc n'en fait pas d'analyse syntaxique (moc ne faisant pas le prétraitement de manière générale).

4.2. Saut des tests unitaires

Un des nombreux changements dans les sources et incompatibles avec Qt 5 se trouve dans le module QtTestLib (un changement dans la macro QSKIP). Dans Qt 4, cette macro prenait deux arguments (Lien 94) ; dans Qt 5, seulement un (Lien 95).

Ceci pose un problème de portabilité. La solution dans KDE est de créer une macro qui prend toujours deux arguments et en passe un sous silence en mode Qt 5. Ceci est fait dans l'optique d'une suppression future lorsque la compilation en mode Qt 4 ne sera plus requise dans KDE.

Une autre solution a été introduite il y a quelque temps (Lien 96), permettant d'utiliser C99 et C++11 (Lien 97). Si vous activez l'option -std=c++11 quand vous compilez votre logiciel, la compatibilité des sources sera restaurée.

4.3. QMetaMethod::signature

Dans les logiciels qui utilisent massivement le système d'introspection de QMetaObject, il est relativement commun d'utiliser l'API QMetaMethod::signature. Dans Qt 4, elle retourne un const char*. Dans Qt 5, cependant, la valeur retournée est construite dynamiquement, elle est donc d'un autre type (cela n'a pas de sens de retourner un pointeur vers une valeur temporaire). Maintenant, la méthode équivalente retourne un type différent (QByteArray) et possède un nom différent (QMetaMethod::methodSignature). Un portage naïf vers la nouvelle méthode peut créer des bogues à l'exécution :

```
// Vieille base de code Qt 4. La variable const char * retournée
// est assignée à une variable locale et utilisée
const char *name = mm.signature();
otherApi(name);

// Nouvelle base de code Qt 5 portée naïvement
const char *name = mm.methodSignature();
// Oh-oh, la variable name est un pointeur. La variable QByteArray
// retournée par la méthode methodSignature a déjà supprimé la donnée.
otherApi(name); // Un plantage s'apprête maintenant à exploser.
```

Le bogue à l'exécution peut être évité en définissant QT_NO_CAST_FROM_BYTEARRAY lors de la compilation du code. Ceci peut et doit être activé dans une base de code Qt 4. La méthode est renommée et la modification du type de la valeur retournée peut être effectuée dans une étape de portage indépendante.

4.4. Modifications des méthodes virtuelles

Un petit nombre de méthodes virtuelles a changé de signature dans Qt 5. Cela ne cause pas d'erreur de compilation durant le portage (excepté dans le cas de modifications de méthode virtuelle pure), mais causera des erreurs à l'exécution quand la méthode « ignorée » ne sera pas exécutée. Un autre exemple est le signal QAbstractItem::dataChanged qui gagne un paramètre.

Il y a plusieurs solutions pour résoudre ceci. Le nouveau standard C++11 a une syntaxe pour indiquer qu'une méthode particulière dans une classe est une méthode ignorée d'une méthode virtuelle dans une des classes de base. L'utilisation de cette syntaxe peut être une étape de préportage provoquant des erreurs durant l'étape actuelle de portage. Les erreurs de compilation sont toujours préférées aux erreurs d'exécution, car elles sont plus simples à trouver (bien sûr, on ne peut pas les éviter).

Une autre option est d'activer les avertissements du compilateur pour trouver le problème. GCC peut afficher les avertissements si une méthode dans une classe dérivée cache une méthode virtuelle dans une classe de base (-Woverloaded-virtual). Cela peut être activé dans votre système de compilation comme une étape de préportage, l'étape de portage peut donc être plus explicite. De plus, comme le programmeur pragmatique l'a enseigné (Lien 98), il faut toujours utiliser les hauts niveaux d'options d'avertissements disponibles avec le compilateur, aussi vous pourrez ajouter ceci à votre système de compilation.

5. Après le portage

Tous les logiciels ont besoin d'être maintenus, pas seulement écrits (ou portés, dans le cas présent): il y a des étapes supplémentaires à réaliser pour terminer le portage de Qt 4 vers Qt 5.

Cela signifie éventuellement qu'il ne sera plus possible de compiler avec Qt 4 et Qt 5 dans la même branche, avec le même code. Avoir un portage terminé avec une base de code Qt 5 et avec tous les tests unitaires qui passent marque le point où il faut commencer à utiliser différentes branches pour les versions basées sur Qt 4 et basées sur Qt 5. Une conséquence de ceci est de permettre de porter le code utilisant les API de Qt 5 et qui est déprécié dans Qt 4.

5.1. Portage des méthodes dépréciées

Les méthodes dépréciées sont des méthodes qui existent dans Qt 5, mais qui sont devenues obsolètes ou ont été remplacées par d'autres méthodes. Dans le cas de Qt 5, une large partie de l'API a été conservée pour garder une compatibilité avec Qt 5, et ainsi faciliter le portage, de

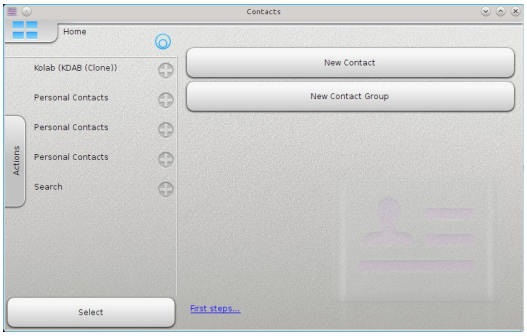
façon similaire avec les méthodes originales de Qt 3 dans Qt 4.

Il est bénéfique de porter également les méthodes dépréciées le plus tôt possible, car il y aura des avertissements durant la compilation, qui peuvent donner plus ou moins de messages d'erreurs selon le compilateur, aussi car les nouvelles API peuvent être plus rapides et utilisent des concepts qui vont de pair avec Qt.

5.2. Portage vers QML

Le portage des interfaces graphiques existantes vers QML est une étape optionnelle dans un portage vers Qt 5 qui peut commencer avant ou après le portage vers Qt 5.

La version de QML qui est disponible dans Qt 4 est maintenant disponible dans Qt 5 sous le nom de Qt Quick 1. Elle est seulement fournie pour une étape de portage et ne recevra pas d'améliorations de performance ou tout autre soin particulier. Le code l'utilisant pourra être porté vers QML 2 (et l'API Qt Quick dans Qt 5) pour bénéficier de la maintenance et de l'amélioration des performances. Le portage vers QML 2 est un cas d'utilisation des noms de classes pertinents dans l'API C++ et met à jour les éléments personnalisés qui sont dessinés. Comme QML 2 utilise une scène graphique plutôt que l'API QPainter (c'est de là que viennent bénéfices en performance), les éléments personnalisés doivent être dessinés en utilisant une API différente.

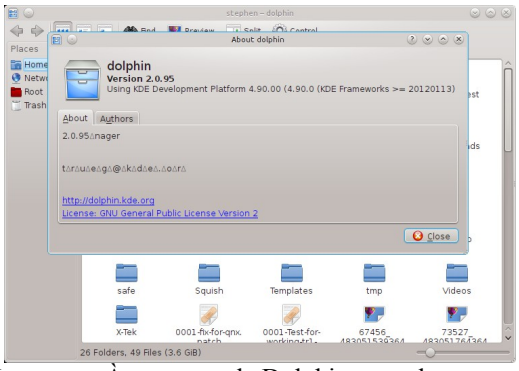


Saisie des contacts dans KDE PIM avec QML 1

6. Conclusion

Ceci est une liste non exhaustive, mais représentative des étapes nécessaires pour le portage de Qt 4 vers Qt 5. Une liste plus longue, mais plus complète des modifications liées au portage fait partie des notes de version de Qt 5.

Il faut aussi noter que cet article de blog cible les étapes requises pour compiler une application avec Qt 5, mais ne parle pas des erreurs d'exécution de portage, qui demandent un réel effort de portage. Quelques applications KDE montrent des bogues subtils qui devront être corrigés :



La page « À propos » de Dolphin avec des erreurs d'encodage

Konqueror avec un bogue mineur dans le menu « Help »

Les modifications sont inscrites dans le fichier de modification et indiquent quelles sont les différences de Qt 5 qui peuvent causer des bogues dans le code porté.

Retrouvez l'article de Stephen Kelly traduit par charlespf en ligne : Lien 99

Les dernières news

Monthly C++ 2012

La communauté du C++ est dynamique et il est possible de trouver tous les mois de nombreuses ressources autour de ce langage. Pour les développeurs, c'est une mine de connaissances importante et une source constante d'inspiration. Cependant, il est parfois difficile de suivre l'ensemble du web et de faire le tri.

C'est pour cette raison que la rubrique C++ a décidé de vous proposer tous les mois, en plus des articles qui vous sont proposés en français, une sélection d'articles en anglais avec un résumé. Si vous connaissez d'autres blogs intéressants à suivre, n'hésitez pas à les proposer ici : Lien 100.

Pour démarrer cette nouvelle série, nous vous proposons cette semaine la sélection d'articles datant de septembre, la semaine prochaine la sélection des mois de juillet et août et la semaine suivante pour les articles plus anciens.

Index des articles

- Décembre 2012 : Lien 101
- Novembre 2012 : Lien 102
- Octobre 2012 : Lien 103
- Septembre 2012 : Lien 104
- Juillet-août 2012 : Lien 105

Que pensez-vous de cette initiative de Developpez.com ?
Quels autres articles en anglais conseilleriez-vous ?

Bonne lecture à tous.

Commentez la news de Guillaume Belz en ligne : Lien 106

Compiler son programme C/C++ en ligne, l'ISO C++ publie des implémentations de Clang, Visual C++ 2012 et GCC, accessibles dans le navigateur

Vous voulez essayer C++, mais vous n'avez pas de compilateur installé sur votre poste ? Vous voulez découvrir une nouvelle fonction de la spécification C++11 ou encore vous voulez comparer les résultats de compilation entre différents compilateurs ?

Voilà quelques situations auxquelles vous pouvez être confronté et qui peuvent rapidement être résolues grâce à un compilateur dans le Cloud.

Plusieurs sociétés ont travaillé sur des implémentations des compilateurs C/C++ accessibles en ligne.

Le comité de normalisation du C++ vient de publier une liste de ces compilateurs online, qui permettent de compiler son programme C++ dans un navigateur, sans avoir besoin d'un compilateur installé sur son poste.

La liste propose les compilateurs en ligne : LiveWorkspace (Lien 107), qui implémente les dernières versions de Clang (3.2) et GCC (4.7.2) ; gcc.godbolt.org (Clang 3.0, GCC 4.5.3 - 4.8.0 bêta, Intel ICC 13.0.1) (Lien 108) ou encore Rise4Fun (Lien 109) qui implémente Microsoft Visual C++ 2012.

LiveWorkspace, par exemple, dispose également des compilateurs pour C# (Mono), Python (PyPy) ou encore Fortran (GFortran).

La liste de compilateurs en ligne de l'ISO C++ : Lien 110

Commentez la news d'Hinault Romaric en ligne : Lien 111

Les derniers tutoriels et articles

Compte rendu Qt Developer Days 2012

Du 12 au 14 novembre 2012 se sont tenus les Qt Developer Days 2012 dans la capitale allemande. Au cours de ces trois jours de conférences, j'ai pu en apprendre plus sur le futur de Qt et sur les bonnes pratiques liées à l'utilisation du framework.

1. Introduction

1.1. Présentation

Les Qt DevDays sont un événement annuel permettant aux développeurs du framework Qt de présenter les nouveautés aux professionnels. C'est aussi l'occasion pour les personnes assistant à l'événement de rencontrer d'autres développeurs Qt, d'assister à des conférences, de discuter avec les développeurs de Qt et experts de la technologie et de découvrir des projets innovants utilisant Qt.

Ainsi, sur trois jours, une cinquantaine de sessions et séminaires ont eu lieu. Pour rappel, le premier jour est une journée de formation afin d'apprendre à utiliser un nouveau module de Qt ou de se remettre à niveau pour

participer aux conférences des deux jours qui suivent.

Ces deux autres journées permettent d'en apprendre plus sur ce qu'allaient être Qt 5, Qt Quick 2, l'évolution envisagée de Qt et son support sur les différentes plateformes du marché, que ce soit les téléphones, les PC ou encore les systèmes embarqués.

1.2. Qt 5 et le futur

Qt 5 approche à grands pas. À l'occasion des Qt Developer Days, la seconde bêta de Qt 5 a été publiée. L'événement a été l'occasion d'en apprendre plus sur les nouveautés de Qt 5 et les changements apportés par cette mouture. Ce n'est pas tout : une description des fonctionnalités principales a été donnée pour les versions Qt 5.1 et Qt 5.2, qui sont prévues pour le courant de l'année 2013.

1.3. Digia reprend les rênes

Le second aspect important de cette année est le transfert de Qt à Digia : Lien 112. Comme vous avez pu le remarquer sur l'affiche, le nom de Nokia n'apparait pas. En effet, après un moment d'incertitude (Lien 113) sur la préparation de l'événement, KDAB et ICS ont repris le flambeau de l'organisation, Digia s'étant joint à eux plus tardivement. Les keynotes du mardi matin nous ont appris que Qt n'a rien subi à cause du transfert de propriétaire, et cela, grâce à l'*open governance*.

Finalement, après une année d'*open governance*, les premiers résultats sont prometteurs et montrent que cette nouvelle gestion fonctionne et apporte des bénéfices au projet Qt.

2. Keynotes

Les keynotes présentent au public les avancées de Qt, les nouveautés et les directions du projet. C'est un moment important, car c'est pendant ces séances que les annonces sont faites.

2.1. Objectifs pour Qt

Tommi Laitinen de Digia a énoncé les objectifs que l'entreprise veut faire atteindre à Qt :

- être le numéro un dans la gestion de plateformes (BlackBerry 10 qui repose en partie sur Qt, mais pas seulement, comme cela est présenté plus loin) ;

- être le numéro un dans l'expérience développeur (meilleurs outils, meilleure documentation, etc.) ;

- être le numéro un dans l'expérience utilisateur (interface fluide, fonctionnalités, performances, etc.) ;

- garder le modèle à double licence (Open Source et commerciale) ;

- renforcer l'écosystème autour de Qt (par exemple, par le développement de partenariats) ;

- garder l'architecture ouverte du Qt Project.

Voir la vidéo : Lien 114

2.2. Développement de Qt

Ensuite, Lars Knoll, mainteneur en chef, est arrivé en scène pour décrire d'un point de vue technique l'évolution arrivant avec Qt 5 et la suite, car Qt ne s'arrête pas à la version 5.

En introduction à sa keynote, Lars a annoncé la disponibilité de la bêta 2 de Qt (Qt 5 est maintenant disponible).

2.2.1. Futur de Qt

Les grandes nouveautés à venir pour l'année 2013 ont été dévoilées. Avec le printemps, c'est le support de BlackBerry qui viendra s'ajouter à Qt 5.1. Plus tard dans l'année viendront se greffer les supports de iOS et d'Android. Comme vous pouvez le constater, l'objectif de Digia pour déployer Qt partout et surtout sur les mobiles prend forme.

2.2.2. Nouveautés de Qt 5

Finalement, Lars Knoll a revu les différentes améliorations de Qt 5 :

- nouvelle abstraction pour le code spécifique aux plateformes avec QPA (pour porter Qt plus facilement) ;
- nouvelle architecture graphique, utilisation optimale du GPU ;
- utilisation du C++11, lorsque le compilateur le permet ;
- signaux/slots revus ;
- utilisation de l'Unicode par défaut (les chaines de caractères sont encodées en UTF-8 par défaut).

Dans les détails, Qt 5 apportera :

- Qt Core :

le support du JSON,

le support des mimetype,

les expressions régulières basées sur PCRE ;

- Qt Network :

QDnsLookup,

un support amélioré de l'IPv6,

un support amélioré du SSL ;

- Qt QML & Qt Quick :

nouveau moteur QML,

améliorations du système de typage,

affichage basé sur un graphe de scène OpenGL,

ajout d'un système de particules,

améliorations dans le système d'import et la gestion des modules ;

- Qt Multimedia :

lecture et enregistrement audio et vidéo,

streaming,

support des caméras,

utilisation des API natives en backend ;

- Qt Webkit :

mise à jour de WebKit apportant le support des dernières fonctionnalités HTML5,

séparation des processus entre WebKit 2 et Qt Quick.

Dans le futur, il est prévu que les composants Qt Quick soient disponibles pour desktops et plateformes tactiles avant l'été prochain. Le système de compilation qmake commence à être revu et son successeur, qbs, sera peut-être disponible avant l'été.
Progressivement, les outils seront améliorés, avec le support complet de plus d'add-ons.

2.2.2.1. Des modules

Qt devient modulaire. Cela permettra aux développeurs d'insérer dans leurs programmes uniquement les morceaux de Qt dont ils ont besoin et de ne pas surcharger l'exécutable final d'éléments inutiles.

Certains modules de Qt 4 ont été dépréciés (mais restent présents dans Qt 5) :

- Qt OpenGL (notamment, car OpenGL est dorénavant intégré dans le cœur de Qt);
- Qt XML ;
- Qt SVG ;
- Qt Script et les outils de scripting.

2.3. Qt Project et l'Open Governance

Finalement, Thiago Maciera a parlé de l'Open Governence. Pendant sa présentation, nous avons pu voir un graphique retraçant les différents commits de l'année sur le Qt Project, par entreprise. Parmi les points forts, on remarque une période où le nombre de commits a fortement diminué, à l'annonce de l'abandon du support de Nokia. Sinon, le développement de Qt est stable et la récente diminution marque le blocage des fonctionnalités en vue de la sortie de Qt 5.

Clairement, la conclusion du graphique et de la keynote est que l'*Open Governance* a permis à Qt de survivre à l'abandon par Nokia. Durant cette année le Qt Project a reçu 25 000 commits (acceptés) de 450 commiters.

Par la suite, Thiago nous a rappelé la facilité avec laquelle il est possible de s'intégrer dans le Qt Project. Grâce à son architecture pyramidale, l'intégration dans le projet suit le niveau d'expertise des participants. En effet, si vous êtes complètement étranger au projet, le plus simple sera de suivre les *mailing lists*. Ensuite, vous saurez comment faire votre premier commit, puis de nouveaux et, si vous êtes assidu, vous pourrez même être promu « approver » (personne approuvant les commits des autres développeurs). Si vous n'en restez pas là et que votre travail continue sur Qt, il vous sera peut-être proposé la place de mainteneur pour votre module dans lequel vous êtes devenu expert. C'est cette architecture qui permet à tout le monde de s'insérer dans le projet et d'y participer.

Vous pouvez télécharger le support de cette keynote : Lien 115.

2.4. BlackBerry 10

Après que les conférences ont été finies, quelques keynotes moins importantes ont eu lieu. Alec Sauncers, Vice President de QNX Software Systems Ltd, nous a parlé de BlackBerry 10 avec ou sans rapport avec Qt.

En effet, la présentation a commencé par quelques faits :

- les revenus des applications BlackBerry sont supérieurs à ceux des autres systèmes ;

- les applications BlackBerry coûtent moins cher à développer.

Ensuite, nous avons pu voir BlackBerry 10 en démonstration. Le système s'inspire de ses concurrents tout en apportant des améliorations qui plaisent à l'œil,

mais qui ne sont pas de première nécessité. En effet, on peut voir pendant la transition, en transparence, les éléments qui vont arriver sur l'écran.

Le clavier pour écrire du texte autocomplète le texte (en devinant les mots - qui s'affichent en transparence sur les lettres du clavier). De plus, il est capable de deviner la langue utilisée par l'utilisateur (énorme avantage si on écrit un texte en plusieurs langues).

RIM fait tout pour avoir énormément d'applications pour son système. Par exemple, il y a quatre façons de développer une application pour BlackBerry 10 :

- C++ (Qt ou Cascades) ;

- HTML 5 ;

- Adobe Air ;

- Android.

Pour les applications Android, il y a tout de même quelques limites. Les applications Jelly Bean ne sont pas encore acceptées. De plus, il ne faut pas que l'application utilise les API payantes de Google (car le support d'Android repose sur la version Open Source).

De plus, RIM organise plusieurs événements pour les développeurs. Des événements techniques un peu partout dans le monde offrant une occasion de rencontrer les professionnels pour avoir du support sur son application.

RIM offrait 100 € pour porter une application durant les Qt Dev Days.

De plus, RIM organise un parrainage (Lien 116) au cours duquel la société assurera que vous gagnez 10 000 dollars si vous développez une application faisant 1000 dollars de revenus.

On comprend très bien que les applications sont une priorité pour la réussite de l'écosystème.

Nous avons aussi pu voir une application Qt Quick avec Box2D (un moteur physique 2D) fonctionner sur l'appareil. Le portage semble très facile, car Qt est parfaitement géré par le système.

Finalement, quelques petits points cités :

- le meilleur support de HTML 5 (que ce soit pour bureau ou pour mobile) ;

- si vous ne respectez pas la vie privée de l'utilisateur, votre application sera rejetée ;

- pour l'instant, les applications GPLv3 ne sont pas acceptées (mais pourraient l'être dans le futur).

Vous pouvez trouver plus d'informations sur le portage d'applications Qt pour le BlackBerry 10 sur le blog de développement : Lien 117.

2.5. Surprises

Cette dernière keynote était plus amusante. Eirik Chambe-Eng (l'ex-président de TrollTech) nous a présenté quelques-uns des *easter eggs* (ajout de code amusant et souvent inutile) de Qt.

Pour ceux qui ont connu les premières années de Qt, le logo était bleu (pendant les six premières années de la bibliothèque).

Dans le fichier Ism de Qt 0.90 (la première version publique), l'équipe avait inclus une typo :

qt-buts@trolltech.no

à la place de :

qt-bugs@trolltech.no

Le chargement des noms des polices de X11 utilisait un nombre magique, qui semble avoir été réutilisé dans GTK.

Q_METHOD est un symbole qui existe dans les fichiers d'entêtes de Qt, mais n'a jamais été utilisé. Celui-ci existe depuis Qt 0.90 et aurait dû être utilisé pour un système de scripting. Malgré sa non-utilisation, le symbole était encore présent dans la première bêta de Qt 5.

Dans Qt 0.90 à 1.2, la fonction écrivant les fichiers XPM (fichier image sous une forme de texte) choisit, comme première couleur, une couleur appelée "Qt".

Dans Qt 0.90 à 1.2, le programme d'exemples *widget* affiche le logo de Qt dans la QList à la ligne 42. Ces versions avaient un style OS/2 (depuis, celui-ci a été retiré, le système étant mort).

Dans Qt 2, jusqu'à 3.2, si on appuie sur Ctrl + Alt, puis 't' 'r' 'o' 'l' 'l' , une boite de dialogue nommée "Egg" s'ouvre.

Dans Qt 4.2, dans la boite de dialogue d'informations du designer, si on maintient le clic sur le logo et que l'on glisse le curseur sur les pixels noirs du logo, un bouton apparait. En cliquant sur le bouton un jeu apparait, affichant un labyrinthe dans lequel on peut collecter les cartes de visite des développeurs.

3. Conférences

3.1. Qt et les API Google

Qt and the Google APIs, **ICS**

Le support de présentation est disponible : Lien 118.

Les API Google sont des services basés sur SOAP destinés principalement aux applications web, mais pas seulement.

ICS a créé et mis à disposition (Lien 119) en licence GPL (LGPL probablement bientôt) des interfaces Qt pour 18 services Google :

- OAuth (authentification, dépendance pour d'autres services) ;

- Tasks (gestion de tâches) ;

- Maps, dont Street View ;

- Latitude (localisation) ;

- Blogger ;

- Calendar (agenda) ;

- Drive (stockage de fichiers dans le cloud) ;

- Freebase (base de connaissances reliée dans un graphe) ;

- Places (infos sur l'environnement de l'utilisateur) ;

- Shopping (informations sur des produits) ;

- Big Query (requêtes) ;

- Predictions (moteur de prédictions, puissant, mais nécessite de définir ses modèles) ;

- Earth (seulement sur Windows).

Attention, certains services sont payants au-dessus d'un plafond d'utilisation (Prediction, par exemple).

Ces interfaces sont disponibles pour Qt en C++ (Linux/Windows/Mac OS X) et QML (N9 et Symbian) ; elles pourraient être intégrées comme module add-on de Qt dans le futur.

3.2. QML pour les applications desktop

QML for desktop apps, par **Michael Wagner** et **Helmut Sedding**

Le support de présentation est disponible : Lien 120.

La conférence avait pour but de nous montrer qu'une application lourde (un logiciel de visualisation et d'édition de chaînes de montage, en 3D) pouvait être réalisée avec QML. En effet, souvent, les développeurs imaginent que QML n'est utile que pour du prototypage ou des applications mobiles, mais cela n'est pas la réalité. L'interface en QML affichait une vue en 3D dans laquelle il était possible de sélectionner, déplacer et modifier les éléments. Une seconde vue, cette fois-ci en 2D, mais toujours en QML permettait de voir la chaine de travail vue de dessus. De même, il était possible de modifier les éléments de la chaîne en temps réel. Celle-ci a été réalisée en se basant sur l'élément basique « Flickable ». Ainsi, plus aucun doute n'est possible, il est tout à fait réalisable de créer une application complexe et de la présenter avec QML.

Afin de réaliser des listes de données filtrables, il est possible de chainer les ProxyModel afin de filtrer des données précédemment filtrées.

3.3. Qt et les services de Cloud

Qt and Cloud Services, par **Sami Makkonen, Digia**

Le support de la présentation est disponible : Lien 121.

La présentation récapitule les différents modèles "cloud", avec des exemples par catégories :

- Saas, *Software as a service*, comme les Google Apps, Dropbox ;

- Baas, *Backend as a service*, comme Firebase, Apigee ;

- Paas, *Platform as a service*, comme Google App Engine, AWS ;

- Iaas, *Infrastructure as a service*, comme Rackspace, AWS.

Plusieurs briques de base pour l'utilisation de services cloud sont déjà disponibles dans Qt : gestion du XML, classes, Network, JSON, SOAP...

Toujours côté Qt, un projet de recherche de l'université de Jyväskylä expérimente l'intégration avec les services Amazon (AWS) et MS Azure pour le stockage de fichiers. Il a été proposé comme projet pour le « playground » Qt.

La session se termine par une présentation d'un projet Baas de Digia : engin.io, qui fournit des services aux applications et aux développeurs : stockage des données, gestion des environnements de développement/test/production.

La démo de codage est assez impressionnante, l'utilisation de base est vraiment très simple. Il utilise en interne (entre autres) MongoDB (pour le stockage) et Amazon S3 (pour le *load balancing*).

En plus de l'API Qt (QML) réglementaire, des API existent pour de nombreux autres langages. Il sort bientôt en bêta, on peut s'inscrire sur le site pour manifester son intérêt.

3.4. Développer avec Qt pour BlackBerry 10

Developing with Qt for the BlackBerry 10, par **Vladimir Minenko, RIM**

Le support de la présentation est disponible : Lien 122.

Cette session présentée par RIM a débuté par un historique de la collaboration entre Qt et RIM. Tout commença donc en novembre 2011 avec une présentation assez particulière annonçant que Qt serait porté sur les appareils RIM. Une année plus tard, l'histoire se concrétise et Qt 4.8.x porté sur BlackBerry 10.

Actuellement, vous pouvez créer une application pour mobiles BlackBerry 10 de quatre manières différentes. L'une d'entre elles utilise le C++ avec Qt ou Cascades. Cascades est une bibliothèque d'interface utilisateur créée par RIM. Sa particularité est de supporter le C++, mais aussi le QML pour la conception de l'interface.

Cascades est basé sur Qt et sera facile à utiliser par les développeurs Qt. La seule particularité est que Cascades utilise son propre graphe de scène (ayant une architecture

client/serveur) et ne permet donc pas d'utiliser l'API de Qt pour le rendu (donc Qt GUI) dans une application Cascades.

Autre petite mauvaise nouvelle : Qt WebKit n'est pour l'instant pas supporté. En effet, RIM a eu quelques difficultés dans le portage et ne pourra pas l'intégrer pour la publication de BlackBerry 10. Toutefois, les développeurs travaillent dessus et son support devrait être réalisé courant 2013.

Parmi les bonnes nouvelles, nous avons pu remarquer que la création d'une application Qt pour BlackBerry est simple. En effet, RIM propose un plug-in pour Qt Creator 2.6.x afin d'avoir un template de projet et de déployer directement son application sur le BlackBerry. Grâce à son NDK, RIM propose aussi un simulateur à travers Qt Creator.

Les applications BlackBerry possèdent quelques particularités, comme celle de nécessiter un fichier XML pour décrire l'application.

Bien entendu, celui-ci sera généré par Qt Creator à travers le plugin de RIM.

RIM tient à la vie privée des utilisateurs. Pour cela, chaque application sera vérifiée sur le store et en plus, chacune d'elles s'exécute dans un espace particulier (*sandbox*).

3.5. QMetaType et QmetaObject en détail

In Depth - QMetaType and QmetaObject, par **Stephen Kelly, KDAB**

Le support de la présentation est disponible : Lien 123.

Le but de cette présentation était de présenter ce qui se cache derrière système de typage de Qt, qui je le rappelle, permet d'enregistrer de nouveaux types dans le cœur de Qt. Stephen Kelly a débuté par nous présenter les améliorations qu'apporte le C++11 pour l'introspection et la reconnaissance des types. Le N3437 propose d'identifier les types avec des chaînes de caractères.

Ensuite, Stephen a détaillé le fonctionnement de QVariant. Celui-ci utilise un entier pour identifier le type contenu. C'est ensuite QMetaType qui permet d'ajouter un nouveau type dans le moteur de Qt et qui fera la correspondance entre les entiers et les types sous forme de chaine de caractères.

Du côté des fonctionnalités disponibles au moment de la compilation (donc sans coût pour l'exécution du programme), il y a le moc et les QObject. Vous interagissez avec le moc à chaque fois que vous utilisez les macros comme Q_OBJECT, Q_PROPERTY…

Avec Qt 5, ce système sera amélioré. La déclaration des métatypes sera automatique et il n'y aura donc plus besoin de Q_DECLARE_METATYPE. Cela est vrai pour toutes les classes héritant de QObject, les conteneurs Qt et les pointeurs intelligents Qt. De plus, il n'y aura plus besoin de qRegisterMetaType dans la plupart des cas. Le code pour enregistrer les types est généré par le moc.

3.6. Créer des compositeurs de fenêtre avec le module Qt Wayland

Creating Window Compositors with the Qt Wayland module, par **Andy Nichols, Digia**

Le support de la présentation est disponible : Lien 124.

Tout d'abord, il faut savoir que Wayland (Lien 125) est un protocole pour les compositeurs (ce qui contient les fenêtres) afin de parler à leurs clients au travers d'une bibliothèque C. Les compositeurs peuvent être des serveurs d'affichage fonctionnant avec Linux, des applications X ou encore un client Wayland lui-même. Les clients peuvent être des applications traditionnelles, des serveurs X (rootless ou fullscreen) ou d'autres serveurs d'affichage.

L'avantage de Wayland est qu'il utilise une mémoire partagée entre les clients afin d'écrire des informations à afficher. Cette mémoire peut être implémentée grâce à un simple système de mémoire partagée ou à travers des buffers GPU (plus rapide encore, car cette méthode évite les coûts d'envoi des données au GPU).

Wayland a l'avantage d'être extensible, rapide, léger et personnalisable. À terme, il pourrait remplacer X11.

Revenons maintenant à Qt. Jusqu'à Qt 4, le système gérant l'affichage des widgets pour les plateformes embarquées était appelé QWS, pour Qt Windowing System. Le problème de ce dernier est qu'il était compliqué d'implémenter de nouvelles plateformes ou même de supporter OpenGL pour gérer l'affichage des fenêtres. C'est pourquoi QWS a été supprimé dans Qt 5. Maintenant, si vous voulez implémenter un nouveau gestionnaire d'affichage, il faudra passer par QPA (Qt Platform Architecture). Qt Wayland propose une implémentation des spécifications Wayland et fournit une méthode pour implémenter un compositeur Wayland, mais aussi des applications clientes s'intégrant aux compositeurs Wayland.

Avec Qt 5.0, pour lancer une application utilisant cette nouvelle plateforme, il suffit d'ajouter :

-platform wayland

Avec cette implémentation, Qt propose une série de nouvelles classes permettant de créer facilement vos propres compositeurs (vos propres interfaces accueillant l'affichage d'autres applications).

WaylandCompositor est la classe principale du compositeur dont il faut hériter afin de créer son compositeur. Dans celle-ci, il est nécessaire de réimplémenter la fonction surfaceCreated() et d'appeler frameFinished() une fois que le rendu des surfaces est terminé.

WaylandSurface représente les surfaces qui seront affichées par les clients. Lors de la réception d'une nouvelle surface avec WaylandCompositor::surfaceCreated(), il est nécessaire de connecter les signaux afin d'être informé des régions qui doivent être dessinées. Il est par contre nécessaire de

vérifier les données reçues à travers les signaux, car elles peuvent être soit des textures OpenGL, soit des segments de mémoire partagée.

Finalement, **WaylandInputDevice**, permet de transmettre les événements utilisateur aux clients (d'envoyer les clics qui sont effectués à partir du compositeur aux applications intégrées dans celui-ci). Ce dernier gère aussi le focus.

Tout cela permet de faire un compositeur accueillant d'autres applications, comme dans la vidéo suivante : Lien 126.

3.7. CMake et Qt

CMake with Qt, par **Stephen Kelly, KDAB**

Le support de la présentation est disponible : Lien 127.

Pour rappel, CMake est un outil permettant de créer les fichiers du projet afin de compiler ce dernier, comme les Makefile, les fichiers de projet pour Visual Studio ou Xcode. Pour ce faire, CMake recherche les dépendances afin d'ajouter les options de compilation nécessaires, les chemins vers les fichiers d'en-tête et les bibliothèques à inclure.

En réalité, CMake a été développé avec trois autres outils :

- CPack, pour créer des paquets de l'application ;
- CTest, pour effectuer des tests unitaires (peut fonctionner avec le système de tests de Qt) ;
- CDash, un serveur pour afficher les résultats des tests.

Même s'il existe qmake, il est possible de compiler ses projets Qt avec CMake. Voici un exemple fichier de configuration de projet Qt pour CMake :

```
find_package(Qt4 REQUIRED QtCore QtGui)

include_directories(${QT_INCLUDES})
add_definitions(${QT_DEFINITIONS})

add_executable(myexe WIN32 main.cpp)
target_link_libraries(myexe
    ${QT_QTCORE_LIBRARIES}
    ${QT_QTGUI_LIBRARIES}
)
```

Il est évident de trouver que les fichiers .pro sont plus simples que les fichiers de Cmake, mais il ne faut pas oublier que ces derniers sont généralistes et peuvent être utilisés pour toute sorte de projets.

Les fichiers de configuration de CMake pour un projet Qt 5 possèdent quelques différences :

```
find_package(Qt5Widgets)

include_directories(${Qt5Widgets_INCLUDE_DIRS})
add_definitions(${Qt5Widgets_DEFINITIONS})
set(CMAKE_POSITION_INDEPENDENT_CODE ON)

add_executable(myexe WIN32 main.cpp)
target_link_libraries(myexe
    ${Qt5Widgets_LIBRARIES}
)
```

Cette lourdeur sera bientôt de l'histoire ancienne. En effet, CMake et son fonctionnement changent. La technique utilisée pour trouver les dépendances des projets (bibliothèques et fichiers d'en-tête nécessaires) sera grandement améliorée et, courant 2013, les fichiers pour CMake ressembleront à ceci :

```
find_package(Qt5Widgets)

add_executable(myexe WIN32 main.cpp)
target_link_libraries(myexe
    Qt5::Widgets
)
```

La bonne nouvelle, c'est que, comme CMake est indépendant de Qt, ces changements fonctionneront aussi pour Qt 4.

Avec Qt 5, il faudra rajouter la ligne suivante afin de réussir la compilation.

set(CMAKE_POSITION_INDEPENDENT_CODE ON)

Par la même occasion, les problèmes de liaisons des projets utilisant QTestLib seront réglés (autour de janvier avec la sortie de CMake 2.8.11).

De plus, la fonction qt4_wrap_cpp pour appeler moc sur les fichiers cpp ne sera plus nécessaire, car CMake détectera de lui-même si le moc est nécessaire directement en examinant les fichiers sources. Par contre, la fonction qt4_wrap_ui sera toujours nécessaire.

Finalement, avec CMake 2.8.10, il sera facilement possible d'ajouter des options aux cibles de compilation avec la syntaxe suivante :

$<$<CONFIG.Debug>:/usr/bin/valgrind>

La variable CONFIG contiendra valgrind uniquement pour la cible Debug (et ne sera donc pas présente en release).

3.7.1. Addons Qt pour tout le monde : KDE Frameworks 5

Qt addons for everyone: KDE Frameworks 5, par **David Faure, KDAB**

Le support de la présentation est disponible : Lien 128.

Au fil du temps, le projet KDE a développé kdelibs, un ensemble de composants communs aux applications KDE. Actuellement, kdelibs est une dépendance monolithique, ce qui crée une frontière entre les applications Qt et les applications KDE.

L'objectif pour KDE 5 est de subdiviser kdelibs en modules avec des interdépendances minimales. Trois types de modules ont été créés :

- fonctionnel : modules de type bibliothèque, pas de dépendance à l'exécution ou à des plug-ins ;

- intégration : modules utilisant des fonctions du système ou les implémentant directement et pouvant avoir des dépendances à l'exécution ;

- solution : technologies complètes avec des dépendances à l'exécution.

Ces modules sont également subdivisés en niveaux de dépendance, le niveau 1 ne dépendant que de Qt, le niveau 2 dépendant du niveau 1 et le niveau 3 dépendant des niveaux 1, 2 et 3.

Quelques exemples de modules :

- fonctionnels, niveau 1 : KArchive, Threadweaver, KIdleTime, Dnssd, KConfig ;

- intégration, niveau 1 : Solid ;

- solution, niveaux 2 et 3 : KIO.

Dans KDE Frameworks 5, les modules fonctionnels et intégration seront utilisables comme des ajouts à Qt, ce qui supprimera effectivement la distinction entre applications « purement » Qt et les applications KDE et le problème d'avoir à choisir entre les deux pour le développement d'une application.

L'aboutissement de ce travail est prévu pour courant 2013, avec un code d'abord basé sur Qt 4, pour séparer le travail de modularisation du travail de portage vers Qt 5 qui sera fait dans la foulée.

3.8. QML efficace : les meilleures pratiques de développement avec Qt Quick

Effective QML: Best practices for Developing with Qt Quick, par **Adenilson Cavalvanti, ICS**

Le **support de présentation est disponible (**partie 1 : Lien 129**,** partie 2 : Lien 130**).**

Durant cette séance, il nous a été possible d'apprendre quelques astuces pour améliorer nos applications. Tout d'abord, il nous a été rappelé que la création d'une application en QML est moins coûteuse en temps de développement et en complexité du code que de faire une interface en C++. Après un rapide récapitulatif du développement en général, différents rappels et astuces ont été présentés.

Dans un code QML, pour les chaines de caractères, il ne faut pas oublier d'utiliser qsTr() pour gérer l'internationalisation. De plus, pour les fichiers de ressources, il ne faut pas passer par les qrc, car le code QML doit être indépendant de l'application. Bien entendu, il est toujours préférable d'enlever les éléments inutiles.

D'autres conseils portaient sur la création des widgets. Pour chacun d'eux, il faut définir une taille minimale. De plus, il est prudent de tester si les propriétés sont définies.

Ensuite, Adenilson Cavalvanti a aussi expliqué que l'approche orientée objet était primordiale. Les propriétés doivent permettre l'exposition de la configuration, d'où la nécessité de créer des interfaces publiques.

De plus, les widgets doivent être encapsulés et il ne faut absolument pas accéder aux éléments par leurs propriétés. Les applications complexes bénéficient d'un contrôleur en C++ pour les traitements de données.

Finalement, il ne faut pas abuser des ListView et penser à utiliser les Repeater. Pour le chargement, il est préférable de permettre un chargement par module, afin d'alléger le démarrage de l'application et surtout, sa consommation en mémoire. N'hésitez pas à penser au chargement asynchrone.

Pour l'optimisation de l'application, il est possible d'écrire des kernels OpenCL en QML. Aussi, il est prudent d'éviter les pointeurs.

3.9. SoDeclarative – une surcouche déclarative pour OpenInventor

SoDeclarative - a declarative wrapper for OpenInventor, **par Helmut Sedding/Michael T. Wagner**

Le support de la présentation est disponible : Lien 131.

IPOPlan (Lien 132) a présenté son logiciel IPO.Log (Lien 133) basé sur Qt 4.8 et QML. Le logiciel contient entre autres une vue 3D en QML (avec Qt 4.8).

Helmut Sedding et Michael T. Wagner nous ont expliqué qu'ils avaient créé une implémentation de OpenInventor, le célèbre graphe de scène de SGI, dans QML. Ainsi, il est aisé de créer de nouveaux objets et scènes avec le langage déclaratif.

Pour cela, ils ont scanné les fichiers sources d'OpenInventor afin de les convertir en vue de leur intégration en QML. Au final, il suffit d'inclure SoDeclarative au début de son fichier pour avoir accès aux différents éléments 3D d'OpenInventor. Il devient donc possible de voir les modifications du code en temps réel dans un viewer QML.

Leur projet est disponible sous licence LGPL sur BitBucket : Lien 134.

3.10. Coder en QML : les outils pour la performance et le débogage

QML Coding, Performance and Debugging: Usage of Tools, par **Aurindam Jana, Digia**

Le support de la présentation est disponible : Lien 135.

Cette présentation énumérait les outils présents dans Qt Creator afin de créer, déboguer et profiler le code QML.

Dans les outils de création d'interfaces en QML, nous pouvons soit utiliser l'éditeur de code, proposant une coloration syntaxique, une barre d'outils spécifique à Qt Quick, des options de refactorisation ou utiliser le designer permettant à un non-programmeur de créer ces interfaces en glissant/déposant les éléments.

L'une des fonctionnalités de refactorisation présentée est la

possibilité de sélectionner un bloc de code QML et de demander à Qt Creator de l'exporter dans un nouveau fichier afin d'en faire un composant réutilisable.

Qt Creator propose différents outils pour déboguer son code en temps réel. Il est possible de définir un point d'arrêt dans le code QML ou C++ et d'observer les valeurs des variables. Il est aussi possible de parcourir l'arbre des éléments Qt Quick et de modifier les propriétés de ce dernier. Ensuite, un ensemble de fonctions pour afficher des informations dans la console est disponible.

Finalement, l'outil de profiling QML a été présenté. Ce dernier permet d'avoir un diagramme du temps passé dans chaque partie du code (affichage, binding, chargement), de voir les boucles de binding (sorte de boucle infinie spécifique au QML), d'avoir un tableau du temps passé dans chacune des fonctions et de savoir quelles sont les fonctions appelantes et les fonctions appelées.

Chaque fois que le profileur indique une fonction, l'éditeur centre sa vue sur le code pour nous montrer le code lié à la fonction.

Pour activer le débogage des applications QML, il faut compiler l'application avec la variable configurée comme suit (dans le fichier .pro) :

CONFIG+=declarative_debug

ou pour Qt 5

CONFIG+=qml_debug

(puisque le module declarative s'appelera "qml" dans Qt 5).

De plus, avec Qt 5 il sera possible d'arrêter le débogueur sur les exceptions JavaScript. Finalement, Qt 5 apportera un profiler QML sous la forme d'une application externe (non liée à Qt Creator). Ce dernier sauvegardera les informations du profiling dans un fichier XML, chargeable dans Qt Creator.

Pour finir, si vous souhaitez déboguer une application QML compilée à l'aide de CMake, les variables à définir sont :

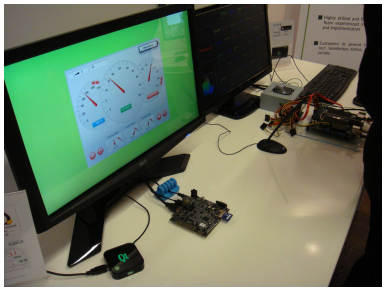
Pour Qt 4 : QT_DECLARATIVE_DEBUG

Pour Qt 5 : QT_QML_DEBUG

4. Exhibition Area

Comme chaque année, l'événement accueille une salle « présentoirs » afin de mettre en avant des projets du monde réel utilisant Qt.

C'est l'occasion de voir l'utilisation de Qt sur des plateformes embarquées :



ou encore dans des Smart TV :

L'espace est aussi utilisé pour montrer des produits aidant les développeurs Qt. C'est ce que font KDAB et Frologic, comme montré ci-dessous :

Ce logiciel portant le nom de « Squish » permet de tester automatiquement vos interfaces utilisateur. Suivant une base de données, le programme remplit les champs de l'application et vérifie que le résultat que vous attendez et bien celui donné par le programme.

Finalement, je voulais aussi montrer ce joli électrocardiogramme en QML qui m'a rappelé le second défi de la rubrique Qt (Lien 136) :

Retrouvez l'article d'Alexandre Laurent en ligne : Lien 137

Liens

Lien 01 : http://philippetulliez.developpez.com/fichiers/AdvancedFilter_v1_0.xls
Lien 02 : http://philippetulliez.developpez.com/tutoriels/advancedfilter/
Lien 03 : http://fauconnier.developpez.com/articles/vba/general/classes/
Lien 04 : http://windows.microsoft.com/fr-FR/skydrive/system-requirements
Lien 05 : http://db.tt/luWcVhHB
Lien 06 : http://loufab.developpez.com/tutoriels/access/cloud_storage_msaccess/
Lien 07 : https://developers.google.com/maps/documentation/android/
Lien 08 : http://www.youtube.com/watch?v=NXwh0h08AkQ
Lien 09 : http://www.developpez.net/forums/d1285614/webmasters-developpement-web/javascript/bibliotheques-frameworks/apis-google/google-met-jour-l-api-google-maps/
Lien 10 : http://www.developpez.net/forums/d1290759/logiciels/solutions-dentreprise/cloud-computing/google-apps-integre-quickoffice-s-attaquer-microsoft-office/
Lien 11 : http://acesyde.developpez.com/tutoriels/android/wifi-android/
Lien 12 : http://www.hibernate.org/
Lien 13 : http://www.h2database.com/html/main.html
Lien 14 : http://www.liquibase.org/
Lien 15 : http://code.google.com/p/mybatis/wiki/Migration
Lien 16 : http://ronnyguillaume.developpez.com/migration-donnees-code-first/
Lien 17 : http://opcoach.developpez.com/articles/eclipse/injection-eclipse4/
Lien 18 : http://netbeans.org/community/releases/73/
Lien 19 : http://www.developpez.net/forums/d1306644/java/edi-outils-java/netbeans/netbeans-7-3-sort-release-candidate-integre-easel/
Lien 20 : http://www.synbioz.com/blog/2012/10/04/comment_fonctionne_les_sessions
Lien 21 : http://synbioz.developpez.com/tutoriels/ruby/middlewares-fondations-rails/
Lien 22 : http://synbioz.developpez.com/tutoriels/ruby/mecanisme-sessions-rails/
Lien 23 : http://pow.cx/
Lien 24 : http://www.synbioz.com/blog/2012/10/10/gestion_de_sous_domaines_avec_rails
Lien 25 : http://matthewhutchinson.net/2010/10/27/rails-3-subdomain-validation-activemodeleachvalidator
Lien 26 : http://synbioz.developpez.com/tutoriels/ruby/gestion-sous-domaines-rails/
Lien 27 : http://www.paulund.co.uk/create-polaroid-image-with-css
Lien 28 : http://paulund.developpez.com/tutoriels/css/images-polaroid/fichiers/demo.html
Lien 29 : http://paulund.developpez.com/tutoriels/css/images-polaroid/
Lien 30 : http://dmouronval.developpez.com/tutoriels/css/fenetre-modale-css3/#LIII-A
Lien 31 : http://debray-jerome.developpez.com/articles/comprendre-les-couleurs-en-css3/#LIV
Lien 32 : http://sohtanaka.developpez.com/tutoriels/javascript/creez-fenetre-modale-avec-css-et-jquery/
Lien 33 : http://debray-jerome.developpez.com/articles/les-selecteurs-en-css3/#LII
Lien 34 : http://dmouronval.developpez.com/tutoriels/css/fenetre-modale-css3/fichiers/
Lien 35 : http://dmouronval.developpez.com/tutoriels/css/fenetre-modale-css3/fichiers/modalbox.zip
Lien 36 : http://dmouronval.developpez.com/tutoriels/css/fenetre-modale-css3/
Lien 37 : http://css.developpez.com/cours/
Lien 38 : http://fireprawn.developpez.com/tutoriels/css/dimensionnement-rem/
Lien 39 : http://www.html5rocks.com/
Lien 40 : http://www.whatwg.org/specs/web-apps/current-work/#constraint-validation
Lien 41 : http://www.whatwg.org/specs/web-apps/current-work/#attr-input-min
Lien 42 : http://www.whatwg.org/specs/web-apps/current-work/#attr-input-max
Lien 43 : http://www.whatwg.org/specs/web-apps/current-work/#attr-input-step
Lien 44 : http://www.whatwg.org/specs/web-apps/current-work/#the-pattern-attribute
Lien 45 : http://www.whatwg.org/specs/web-apps/current-work/#attr-input-required
Lien 46 : http://www.whatwg.org/specs/web-apps/current-work/#attr-fe-maxlength
Lien 47 : http://www.whatwg.org/specs/web-apps/current-work/#attr-input-type
Lien 48 : http://caniuse.com/#feat=form-validation
Lien 49 : http://www.whatwg.org/specs/web-apps/current-work/#constraint-validation-api
Lien 50 : http://www.whatwg.org/specs/web-apps/current-work/#category-submit
Lien 51 : http://www.whatwg.org/specs/web-apps/current-work/#dom-cva-validity
Lien 52 : http://www.whatwg.org/specs/web-apps/current-work/#validitystate
Lien 53 : http://tjvantoll.com/2012/10/17/maxlength-constraint-validation-oddities/
Lien 54 : https://github.com/necolas/normalize.css/pull/124
Lien 55 : https://github.com/necolas/normalize.css
Lien 56 : http://jsfiddle.net/tj_vantoll/a533m/
Lien 57 : https://bugs.webkit.org/show_bug.cgi?id=48491
Lien 58 : https://www.w3.org/Bugs/Public/show_bug.cgi?id=10923
Lien 59 : http://www.alistapart.com/articles/inline-validation-in-web-forms/
Lien 60 : http://blog.oldworld.fr/index.php?post/2011/05/Improving-HTML5-Forms-user-experience-with-moz-ui-invalid-and-moz-ui-valid-pseudo-classes
Lien 61 : http://www.w3.org/TR/selectors4/#user-pseudos
Lien 62 : http://remysharp.com/2010/10/08/what-is-a-polyfill/
Lien 63 : https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-Browser-Polyfills
Lien 64 : http://afarkas.github.com/webshim/demos/index.html
Lien 65 : http://afarkas.github.com/webshim/demos/demos/webforms.html
Lien 66 : http://www.alistapart.com/d/forward-thinking-form-validation/enhanced_2.html
Lien 67 : http://tjvantoll.com/2012/08/05/html5-form-validation-showing-all-error-messages/
Lien 68 : https://github.com/ryanseddon/H5F
Lien 69 : http://dmouronval.developpez.com/tutoriels/javascript/api-contrainte-validation/
Lien 70 : http://andnotor.developpez.com/tutoriels/delphi/envoyer-chaines-postmessage/fichiers/MessagesEx.zip
Lien 71 : http://andnotor.developpez.com/tutoriels/delphi/envoyer-chaines-postmessage/
Lien 72 : http://andnotor.developpez.com/tutoriels/delphi/fasturldownload/
Lien 73 : http://qt.developpez.com/actu/51460/Qt-5-0-1-est-disponible-avec-le-support-de-MinGW-4-7-et-une-incompatibilite-mineure-du-plug-in-Qt-Multimedia/

Lien 74 : http://qt.gitorious.org/qt-creator/qt-creator/blobs/2.6/dist/changes-2.6.2
Lien 75 : http://qt-project.org/downloads
Lien 76 : http://www.developpez.net/forums/d1304673/c-cpp/bibliotheques/qt/edi/qt-creator/sortie-qt-creator-2-6-2-a/
Lien 77 : http://www.youtube.com/watch?v=FIs5YqzS4Bc
Lien 78 : https://qt.gitorious.org/qt-labs/qt5-launch-demo
Lien 79 : http://www.youtube.com/watch?v=vhWS_bN-T3k
Lien 80 : http://qt-project.org/wiki/Qt500KnownIssues
Lien 81 : http://qt-project.org/contribute
Lien 82 : http://qt-project.org/downloads
Lien 83 : http://www.developpez.net/forums/d1209764-2/c-cpp/bibliotheques/qt/sortie-qt-5-0-1-a/#post7039286
Lien 84 : http://www.developpez.net/forums/d1209764-4/c-cpp/bibliotheques/qt/sortie-qt-5-0-1-a/#post7104658
Lien 85 : http://www.kdab.com/
Lien 86 : http://www.kdab.com/porting-from-qt-4-to-qt-5/
Lien 87 : http://marcmutz.wordpress.com/effective-qt/subclassing/
Lien 88 : http://steveire.wordpress.com/2011/03/16/implementing-qvariantqmetatype-features-with-template-tricks/
Lien 89 : http://doc.qt.nokia.com/4.7-snapshot/qws.html
Lien 90 : http://comments.gmane.org/gmane.comp.lib.qt.user/231
Lien 91 : http://qt.gitorious.org/qt/qtbase/blobs/HEAD/dist/changes-5.0.0
Lien 92 : http://thread.gmane.org/gmane.comp.lib.qt.devel/4450/focus=4456
Lien 93 : http://lxr.kde.org/source/kde/kdelibs/kdecore/util/kexportplugin.h
Lien 94 : http://qt-project.org/doc/qt-4.8/qtest.html
Lien 95 : http://qt-project.org/doc/qt-5.0/qtest.html
Lien 96 : https://codereview.qt-project.org/
Lien 97 : http://en.wikipedia.org/wiki/Variadic_macro
Lien 98 : http://books.google.de/books?id=5wBQEp6ruIAC&pg=PA92&lpg=PA92&dq=pragmatic+programmer+compiler+warning+levels+as+high+as+possible&source=bl&ots=n4ixkzgPp_&sig=RiPRr9EbVoB5U6G4tRIuucYjyF4&hl=en&sa=X&ei=83vPT4vsJJDJswabprTDCg&redir_esc=y
Lien 99 : http://kdab.developpez.com/tutoriels/porter-qt4-qt5/
Lien 100 : http://www.developpez.net/forums/d1234295/c-cpp/cpp/communaute/blogs-traitant-cpp/
Lien 101 : http://www.developpez.net/forums/d1268001/c-cpp/cpp/monthly-cpp/#post7057645
Lien 102 : http://www.developpez.net/forums/d1268001/c-cpp/cpp/monthly-cpp/#post7009178
Lien 103 : http://www.developpez.net/forums/d1268001/c-cpp/cpp/monthly-cpp/#post6966734
Lien 104 : http://www.developpez.net/forums/d1268001/c-cpp/cpp/monthly-cpp/#post6923033
Lien 105 : http://www.developpez.net/forums/d1268001/c-cpp/cpp/monthly-cpp/#post6933175
Lien 106 : http://www.developpez.net/forums/d1268001/c-cpp/cpp/monthly-cpp-2012-a/
Lien 107 : http://liveworkspace.org/
Lien 108 : http://gcc.godbolt.org/
Lien 109 : http://rise4fun.com/vcpp
Lien 110 : http://isocpp.org/blog/2013/01/online-c-compilers
Lien 111 : http://www.developpez.net/forums/d1305328/c-cpp/outils-c-cpp/compiler-programme-c-cpp-ligne/
Lien 112 : http://qt.developpez.com/actu/47705/Premier-jour-de-Qt-chez-Digia-apres-son-rachat-de-Nokia-le-framework-restera-disponible-sous-GPL-et-LGPL/
Lien 113 : http://qt.developpez.com/actu/46153/De-l-avenir-des-Qt-Developer-Days-toujours-pas-d-annonce-officielle-la-communaute-reprend-les-choses-en-main/
Lien 114 : http://www.youtube.com/watch?v=o1PzIBDOw4w
Lien 115 : https://qtconference.kdab.com/sites/default/files/slides/Project_that_brings_Qt.pdf
Lien 116 https://developer.blackberry.com/builtforblackberry/commitment/
Lien 117 : http://devblog.blackberry.com/
Lien 118 : https://qtconference.kdab.com/sites/default/files/slides/QtGoogleAPIs.pdf
Lien 119 : http://www.ics.com/technologies/qt_google_apis/
Lien 120 : https://qtconference.kdab.com/sites/default/files/slides/QML_for_Desktop_Applications_Slides_Publish.pdf
Lien 121 : https://qtconference.kdab.com/sites/default/files/slides/Qt_DevDays_2012_Sami_Makkonen.pdf
Lien 122 : https://qtconference.kdab.com/sites/default/files/slides/Developing_with_Qt_for_the_BlackBerry_10-Berlin.pdf
Lien 123 : https://qtconference.kdab.com/sites/default/files/slides/devdays_Observe_Your_Neighbors.pdf
Lien 124 : https://qtconference.kdab.com/sites/default/files/slides/QtWayland%20Presentation.pdf
Lien 125 : http://wayland.freedesktop.org/
Lien 126 : http://www.youtube.com/watch?v=_FjuPn7MXMs
Lien 127 : https://qtconference.kdab.com/sites/default/files/slides/cmake.pdf
Lien 128 : https://qtconference.kdab.com/sites/default/files/slides/Qt_Addons_KDE_Frameworks.pdf
Lien 129 : https://qtconference.kdab.com/sites/default/files/slides/adenilson_cavalcanti_devdays12_part1.pdf
Lien 130 : https://qtconference.kdab.com/sites/default/files/slides/adenilson_cavalcanti_devdays12_part2.pdf
Lien 131 : https://qtconference.kdab.com/sites/default/files/slides/SoDeclarative_OpenInventor_QML_wrapper.pdf
Lien 132 : http://www.ipoplan.de/
Lien 133 : http://www.ipoplan.de/index.php?module=pagemaster&func=viewpub&tid=1&pid=72&lang=en
Lien 134 : https://bitbucket.org/sodeclarative
Lien 135 : https://qtconference.kdab.com/sites/default/files/slides/devdays_berlin_Usage_QML_Tools.pdf
Lien 136 : http://qt.developpez.com/defis/02-tablette-hopital/
Lien 137 : http://qt.developpez.com/evenement/2012-devdays/reportage/