Developpez

Le Mag

Édition de décembre - janvier 2012/2013.
Numéro 43.
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

MS Office	Page 2
CSS	Page 4
(X)HTML	Page 12
JavaScript	Page 15
Qt	Page 20
C++	Page 22
2D/3D/Jeux	Page 29
Perl	Page 36
Java	Page 44
Dev. Web Java	Page 53
NetBeans	Page 54
Eclipse	Page 55
Android	Page 62
Liens	Page 70

Éditorial

Encore une année où votre magazine préféré vous accompagne et vous fait partager le meilleur de nos articles sur toutes vos technologies préférées.

Profitez-en bien !

La rédaction

Article CSS

Les filtres CSS3 : modifier le HTML ou les images en CSS

Les filtres CSS3 vous permettent d'appliquer, à des éléments HTML ou des images, beaucoup d'effets, nous allons voir précisément comment ils vont fonctionner.

par **Didier Mouronval**
Page 4

Article 2D/3D/Jeux

Faire un jeu vidéo : par où commencer ?

Vous avez une idée de jeu ou vous débutez en programmation et souhaitez faire un jeu, alors cet article est un excellent point de départ pour vous.

par **Alexandre Laurent**
Page 29

Le VBA qu'est ce que c'est ?

L'objectif de ce tutoriel est de préciser, pour le débutant, la nature du VBA qui est souvent confondu avec le Visual Basic (VB) ou considéré comme un langage indépendant, ce qu'il n'est pas.

1. Introduction

Mon ami, Benoît Lecodeur, m'a dit qu'il était devenu passionné depuis qu'il utilisait les macros pour automatiser des tâches, mais qu'il comprenait mal ce qu'était le VBA, pouvait-il l'utiliser seul ? Était-ce le nom du Visual Basic employé par les utilisateurs d'Office ? Etc.
Benoît Lecodeur n'étant pas un professionnel du développement et ne comptant pas le devenir, je décidai de lui faire un topo à sa portée pour clarifier les points essentiels et cadrer les choses les plus importantes.

2. Préambule

Ceci étant un tutoriel pour débutant, nous emploierons des notions générales sur lesquelles les puristes voudront bien fermer les yeux, de même que s'il n'est pas question de négliger les frottements pour les calculs nécessaires à l'envoi de satellites dans l'espace, c'est courant de le faire à des fins pédagogiques en terminale pour enseigner les bases de la physique.
On décrira donc, ici par exemple, le VBA comme un langage orienté objet sans approfondir la chose.

3. VBA l'origine : Les macro-commandes

Le VBA n'est pas apparu parallèlement avec les premières versions d'Office Windows. À l'origine, il y avait les applications Microsoft Office qui disposaient de macro-commandes classiques comme beaucoup de programmes (Lotus 1-2-3 et Multiplan (source Microsoft : Lien 01) en possédaient déjà, entre autres).

Une macro-commande est une suite d'instructions qui peut s'exécuter à l'aide d'une commande unique (clavier ou souris) et qui a été obtenue via l'enregistreur de macro-commandes. Celui-ci, une fois déclenché mémorise les actions effectuées par l'utilisateur jusqu'à ce qu'on le stoppe. Il mémorise ces actions dans un miniprogramme nommé macro-commande.

La particularité des programmes informatiques est qu'on peut généralement avoir accès aux lignes de code ainsi créées, ce qui n'est pas le cas des macro-commandes obtenues par le même moyen dans des appareillages courants tels que les télécommandes d'appareils électroniques, certaines manettes de jeux vidéo ou tout simplement la mémorisation d'un numéro de téléphone dans le répertoire d'un téléphone…
Les macro-commandes ont eu un succès considérable car elles permettaient d'automatiser des tâches récurrentes et fastidieuses relativement facilement et faisaient, ainsi, gagner énormément de temps et de fiabilité.
Le langage VBA en est la suite logique et permet d'aller beaucoup plus loin, tout en permettant à celui qui le veut de se cantonner à l'enrichissement des macros obtenues via l'enregistreur de macros disponible dans certaines applications.

4. Les macro-commandes dans Office

Les macros étaient très utilisées dans les tableurs qui ont été le vecteur principal de leur développement et de celui du VBA.
Elles ont été intégrées en premier dans Excel de 1992 (N° 4) puis dans Excel 93 (N° 5) qui a été le premier programme Microsoft voyant apparaître le langage VBA (voir définition par la suite), ces deux versions utilisaient des feuilles de classeurs spécifiques pour stocker les macros dites XL4 (nom issu de la dernière version d'Excel avant l'ère VBA, elles ont été conservées depuis Excel 95 pour des raisons de compatibilité).
Word 95(N° 6) incluait lui WordBasic, un langage de macros spécifique à Word, Access possédait quant à lui Access Basic (tous ces langages étant incompatibles), etc.
Office 97 avec Excel 97 (N° 8) et Word 97, institutionnalise le VBA en l'incorporant à tous les logiciels de la suite Office comme langage de programmation et, surtout, en lui attribuant un éditeur de code à part entière : le VBE.
Il est à noter que les versions d'Excel 2010 d'Office sont toujours compatibles avec les macros XL4 que l'on peut appeler via le VBA pour des contextes très particuliers. Les macros XL4 recèlent des fonctions qui n'ont pas leurs équivalents directs en VBA, Microsoft a entrepris de compléter le VBA à ce niveau dans Excel 2010 pour permettre une migration complète des anciennes applications tournant encore. Aussi peut être pour permettre l'éradication de l'utilisation des macros XL4 ?
À partir d'Office 97, Microsoft a donc voulu homogénéiser les langages de macro-commandes entre les différentes applications et en développer la puissance pour permettre aux utilisateurs de programmer carrément des codes puissants au travers d'un langage « embarqué ».

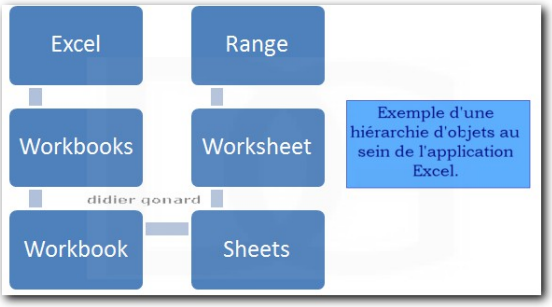
5. Le VBA

Plutôt que de réinventer la poudre, ils ont fort opportunément utilisé les bases de Visual Basic (un langage de programmation à part entière de Microsoft) pour créer le VBA, sigle de **Visual Basic for Application**.

D'où les confusions nombreuses entre les deux langages, alors que, s'il y a de nombreuses similitudes (orientation objet et filiation de VBA), les différences dans leur emploi sont conséquentes.

Le VBA ne peut exister qu'au sein d'une application hôte.

On ne peut accéder à son éditeur de façon séparée et celui-ci n'est accessible qu'au travers de l'application qui l'héberge.
Le VBA est basé sur la manipulation d'objets qui représentent les éléments principaux de l'application employée et d'interfaces de pilotage. Ce modèle objet est organisé selon un modèle hiérarchique d'éléments tels des matriochkas.
À la base, pour tous les programmes supportant le VBA, il y a l'objet Application qui représente le programme visé et sert de conteneur aux objets particuliers à ce programme.
On manipule ces objets par programmation via leurs propriétés (teinte d'une couleur par exemple), leurs méthodes (ajout d'un élément par exemple) et leurs événements (réaction à un clic de souris par exemple).



Les objets sont référencés au sein de bibliothèques d'objets qui permettent de disposer de toutes leurs caractéristiques selon l'environnement où l'on se trouve. On peut, à travers elles, piloter aisément une application supportant le VBA à partir d'une autre. Elles permettent aussi à d'autres langages objet de piloter Office (comme Visual Basic entre autres).
L'intérêt du VBA est aussi que la façon de le manipuler sera identique sur le fond pour tous les programmes qui le supportent. Il suffira de s'imprégner du modèle objet de l'application voulue pour la piloter.
Le fait que le VBA soit lié à une application entraîne aussi, qu'avant toutes choses, il est nécessaire de savoir manipuler l'application en question avant de se lancer dans la construction d'un programme qui la concerne.
Le VBA, comme tout langage, possède une syntaxe. Mais, en tant que langage de programmation, il ne tolère aucune erreur quant à celle-ci. Un espace omis ou mal placé, un point à la place d'une virgule et ce sera le blocage du programme (l'éditeur VBE nous aide bien sur ce point). C'est pourquoi, sur les forums d'entraide, les contributeurs paraissent si rigoristes. Le meilleur moyen de gagner du temps est d'étudier, en premier lieu, les bases du langage au travers de formations, de livres ou de tutoriels disponibles, ce qui est, de plus, plus agréable que de passer son temps à gérer des erreurs basiques.

Une autre particularité du VBA, qui découle de ce qu'on a déjà décrit, est que le code qui est produit est stocké dans le corps d'un fichier de l'application qui a servi au développement et non dans un fichier indépendant. Ceci explique aussi le fait qu'il est impossible de faire un programme exécutable indépendant en VBA. Le code produit au moyen du VBA sera toujours exécuté au travers du programme hôte (celui utilisé pour son écriture).
Une autre conséquence de la particularité du VBA en tant que langage hébergé est qu'au niveau de la documentation, des livres, des forums, vous trouverez rarement quelque chose ciblé VBA en général, mais toujours VBA pour XXXX (telle application).

Il n'en demeure pas moins qu'une fois ce concept de base maîtrisé, vous pourrez tout à fait consulter avec bonheur la littérature VBA liée à une autre application que celle que vous utilisez pour peu que le sujet traite de points qu'elles ont en commun ou qui font partie de ce qu'on pourrait considérer comme un tronc général du VBA.
Il conviendra toutefois d'utiliser en parallèle une alliée précieuse qui se nomme l'Aide VBA liée à l'application ! Celle-ci vous permettra généralement de prendre connaissance des subtilités diverses qui peuvent exister pour un élément apparemment identique dans divers contextes.

6. Répercussions du VBA sur les formats de fichiers Office

Si pendant longtemps les fichiers Office contenant du code ne se distinguaient pas de façon précise, l'activité mal intentionnée des créateurs de virus, qui en ont tiré parti, a renforcé la création de filtres de sécurité par rapport à leur emploi, allant crescendo jusqu'à l'apparition de formats de fichiers particuliers à partir d'Office 2007.
À partir d'Office 2007, les fichiers comportant du code VBA ont une extension bien distincte à tel point que le fait d'enregistrer un fichier avec code sous un format de fichier ordinaire entraînera la disparition totale des tous les modules de code.
Présentation des nouvelles extensions de noms de fichiers par Microsoft : Lien 02

7. Le VBA en dehors d'Office Microsoft

D'autres applications, pas forcément de chez Microsoft, ont aussi opté pour le langage VBA pour permettre de programmer leur interfaçage telles que : AutoCad, MicroStation (CAO), Catia V5-6 (Dassault / CAO), Wordperfect Office X5 (Corel), CorelDraw (Corel), Visio, Solidworks (CAO), etc.

8. Le VBA en dehors d'Office Windows

Le VBA existe aussi dans certaines versions de Mac Office, la compatibilité entre les versions Windows et Mac étant limitée principalement par les différences de syntaxe des systèmes (séparateurs de chemins par exemple) et les modèles objets (contrôles Active X etc.).

9. Conclusions

Connaître ces quelques notions de départ vous permettra de savoir dans quel cadre vous vous situez, de ne pas partir sur des choses irréalisables (.exe en VBA) ou mieux, d'imaginer des choses plus puissantes comme l'interaction entre plusieurs applications supportant le VBA.
Aussi et surtout, savoir, pour ceux qui utilisent les moteurs de recherche, distinguer sur quel genre de site vous avez atterri : Visual Basic, VBA pour l'application que vous cibliez ou VBA pour une autre application.

Retrouvez l'article de Didier Gonard en ligne : Lien 03

Les filtres CSS3 : modifier le HTML ou les images en CSS

Les filtres CSS3 sont une récupération intéressante de SVG. Ils vous permettent d'appliquer, à des éléments HTML ou des images, des flous, de la luminosité et beaucoup d'autres effets. Dans cet article, nous allons voir précisément comment ils vont fonctionner.

Cet article a été traduit avec l'aimable autorisation de InsertHTML (Lien 04). L'article original : CSS3 Filters: Altering HTML and Images with just CSS (Lien 05) peut être vu sur le site de InsertHTML.

1. Comment cela fonctionne

Avec du simple CSS, nous pouvons appliquer des effets assez élaborés. Ils sont supposés s'appliquer aussi bien à des images qu'à des éléments HTML, mais manifestement, le support des navigateurs est variable. La propriété utilisée pour gérer cela est filter.

```
filter: filter(value);
```

Comme vous devez vous en douter, les préfixes vendeurs sont requis. **Pour le moment** (N.D.T : article original publié en juin 2012)**, seul** -webkit- **(Chrome et Safari) reconnaît ces propriétés.**

```
-webkit-filter: filter(value);
-moz-filter: filter(value);
-o-filter: filter(value);
-ms-filter: filter(value);
```

2. Les filtres

Il existe différents types de filtres, pour permettre d'avoir une bonne idée de ce qu'ils réalisent, nous les verrons individuellement. Différents filtres peuvent être appliqués, séparés par un espace, par exemple brightness et blur :

```
filter: blur(5px) brightness(0.5);
```

Il existe certains filtres dont je ne parlerai pas car ils peuvent être obtenus avec d'autres propriétés CSS, comme l'opacité et les ombres portées.
Voici l'image originale sur laquelle nous allons appliquer les filtres :

Je vous montrerai la photo originale (la première) et ce à quoi devrait ressembler l'application du filtre (la seconde), vous pourrez voir la différence même si votre navigateur ne reconnaît pas les filtres.

3. Flou

Vous avez toujours voulu appliquer un flou gaussien sur une image ou du texte uniquement en CSS ? Avec les filtres, c'est désormais possible ! Le flou est mesuré en pixels, vous pouvez donc faire quelque chose comme ça :

```
/* Préfixes vendeurs */
-webkit-filter: blur(5px);
-moz-filter: blur(5px);
-o-filter: blur(5px);
-ms-filter: blur(5px);

filter: blur(5px);
```

4. Luminosité

La luminosité est comprise entre zéro et un, 1 correspondant à une luminosité maximale (blanc) et 0 à la valeur originale.

```
/* Préfixes vendeurs */
-webkit-filter: brightness(0.2);
-moz-filter: brightness(0.2);
-o-filter: brightness(0.2);
-ms-filter: brightness(0.2);

filter: brightness(0.2);
```

5. Saturation

Mesurée en pourcentage avec le mot clé saturate().

```
/* Préfixes vendeurs */
-webkit-filter: saturate(50%);
-moz-filter: saturate(50%);
-o-filter: saturate(50%);
-ms-filter: saturate(50%);

filter: saturate(50%);
```

6. Rotation de teinte

Cet effet est assez intéressant. Il vous permet d'altérer les teintes par rotation (imaginez une roue colorimétrique que vous pouvez faire pivoter). La valeur est mesurée en degrés.

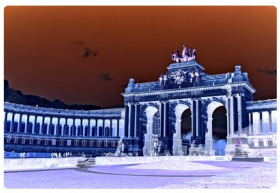
```
/* Préfixes vendeurs */
-webkit-filter: hue-rotate(20deg);
-moz-filter: hue-rotate(20deg);
-o-filter: hue-rotate(20deg);
-ms-filter: hue-rotate(20deg);

filter: hue-rotate(20deg);
```

7. Contraste

Mesuré aussi en pourcentage. 100 % est la valeur initiale, 0 % va créer une image entièrement noire. Toute valeur au-delà de 100 % ajoute du contraste.

```
/* Préfixes vendeurs */
-webkit-filter: contrast(150%);
-moz-filter: contrast(150%);
-o-filter: contrast(150%);
-ms-filter: contrast(150%);

filter: contrast(150%);
```

8. Inversion



La mesure du taux d'inversion souhaité sur l'image. Étonnamment, webkit n'accepte pas, à l'heure actuelle, les valeurs inférieures à 100 %.

```
/* Préfixes vendeurs */
-webkit-filter: invert(100%);
-moz-filter: invert(100%);
-o-filter: invert(100%);
-ms-filter: invert(100%);

filter: invert(100%);
```

9. Nuances de gris

Ici encore, la mesure correspond au niveau de gris que vous souhaitez obtenir, de 0 % à 100 %.

```
/* Préfixes vendeurs */
-webkit-filter: grayscale(100%);
-moz-filter: grayscale(100%);
-o-filter: grayscale(100%);
-ms-filter: grayscale(100%);

filter: grayscale(100%);
```

10. Sépia

Le sépia est certainement très utile si vous souhaitez poster sur Instagram, mais je suppose que dans ce cas, vous n'aurez pas besoin de CSS. Comme pour les nuances de gris ou l'inversion, la valeur correspond au taux de sépia que vous souhaitez ajouter, de 0 % (image initiale) à 100 % (sépia complet).

```
/* Préfixes vendeurs */
-webkit-filter: sepia(100%);
-moz-filter: sepia(100%);
-o-filter: sepia(100%);
-ms-filter: sepia(100%);

filter: sepia(100%);
```

Retrouvez l'article de Johnny Simpson traduit par Didier Mouronval en ligne : Lien 06

Utiliser le modèle des boites CSS - Pour simplifier le dimensionnement des éléments

Dans ce tutoriel, nous allons voir une nouvelle méthode pour dimensionner des éléments HTML en CSS.

1. Traduction

Ce tutoriel est la traduction la plus fidèle possible du tutoriel original de Paul Underwood (Lien 07), Use Box Model For Easier Widths : Lien 08.

2. Introduction

Dans cet article nous allons voir une petite astuce pour modifier la taille des éléments HTML en CSS. Il y a plusieurs choses qui définissent la taille d'un élément comme sa largeur, les marges intérieures ou encore les bordures.
Taille totale = largeur + marges intérieures + bordures.
Cela signifie que si la largeur de votre élément est de 500px avec une marge intérieure de 50px et une bordure de 10px, la taille totale sera alors de 620px.
620px = 500px + (50px * 2) + (10px * 2).

Ce calcul complique un peu la mise en page lorsqu'elle est un peu complexe, si vous souhaitez modifier les marges intérieures ou encore les bordures à l'avenir, il faudra également modifier la largeur de l'élément.

3. La propriété « box-sizing »

Il existe une propriété CSS box-sizing qui peut simplifier ce calcul. En effet en utilisant box-sizing avec la valeur border-box les marges intérieures ainsi que les bordures seront incluses dans la largeur de votre élément. Cela change donc le calcul présenté ci-dessus.

```
div {
        box-sizing: border-box;
        }
```

La taille totale est donc maintenant l'addition de la largeur, des marges intérieures et des bordures de l'élément.
Taille totale = (largeur - marges intérieures - bordures) + marges intérieures + bordures.
Si nous reprenons le même exemple, un élément dont la largeur est de 500px avec des marges intérieures de 50px et une bordure de 10px aura une taille totale de 500px.
500px = 380px + (50px * 2) + (10px * 2).

4. Appliquer cette méthode à tous les éléments

Si vous préférez cette façon de procéder, il est possible de l'appliquer automatiquement à tous vos éléments.

```
* {
     -moz-box-sizing: border-box;
     -webkit-box-sizing:
     border-box; box-sizing: border-box;
}
```

Vous penserez peut-être qu'utiliser le sélecteur * ralentira le chargement du CSS. Paul Irish a mis en évidence après une série de tests que l'utilisation de ce sélecteur ajoute une charge de 4 % seulement, ce qui est négligeable.

5. Support des navigateurs

La propriété box-sizing est supportée par tous les principaux navigateurs.
Internet Explorer la supporte depuis la version 8.
Firefox la supporte depuis la version 2 mais uniquement avec le préfixe -moz.
Chrome la supporte en version préfixée (-webkit) jusqu'à la version 9, et en version non préfixée pour les versions supérieures.
Pour rester au courant des évolutions du support des navigateurs, vous pouvez consulter ce lien : Lien 09.

6. Conclusion

Nous avons vu dans ce tutoriel une nouvelle méthode pour définir la taille de vos éléments HTML en CSS. Cette méthode simplifie les calculs lors d'une mise en page complexe, en intégrant les marges intérieures et les bordures dans la taille définie en CSS.

Retrouvez l'article de Paul Underwood traduit par Sébastien Germez en ligne : Lien 10

Menu fixe avec un effet de slide en CSS3 - Pour avoir une navigation simple et fluide

Dans ce tutoriel nous allons voir comment créer un menu de navigation toujours visible avec un effet de slide au passage de la souris.

1. Traduction

Ce tutoriel est la traduction la plus fidèle possible du tutoriel original de Paul Underwood (Lien 07), How To Create A CSS Slide Out Fixed Navigation : Lien 11.

2. Introduction

Aujourd'hui nous allons apprendre comment rendre la navigation de vos utilisateurs plus facile en créant un menu fixe toujours visible. Mais pour que ce menu ne gêne pas le contenu de chaque page, on masquera la moitié des images utilisées pour le menu. Les images ne s'afficheront entières qu'au survol de la souris.

3. Le code HTML

Pour commencer, nous allons écrire le code HTML nécessaire à la navigation en utilisant le tag <ul> pour créer une liste.

```
<ul class="nav">
<li><a href="#home">
        <img src="./images/home.png" />
    </a></li>
<li><a href="#about">
        <img src="./images/about.png" />
    </a></li>
<li><a href="#portfolio">
        <img src="./images/portfolio.png" />
    </a></li>
<li><a href="#blog">
        <img src="./images/blog.png" />
    </a></li>
<li><a href="#services">
        <img src="./images/services.png" />
    </a></li>
<li><a href="#contact">
        <img src="./images/contact.png" />
    </a></li>
</ul>
```

Chaque item de la liste contient un lien (tag <a>) et chaque lien contient une image. Comme l'exemple ne porte que sur une seule page, les différents liens sont des ancres qui renvoient vers des id d'éléments sur la même page.

4. Le code CSS

La partie la plus importante de cet article concerne le code CSS car il va nous permettre de fixer le menu ainsi que de lui appliquer le style voulu.
Tout d'abord nous allons écrire le code qui permettra de créer la barre de navigation fixe en utilisant la classe CSS nav.

```
ul.nav{
        list-style:none;
        padding:0;
        margin:0;
        position:fixed;
        left:-30px;
        top:140px;
}
```

Ce code permet de ne pas utiliser le style des listes par défaut (puces) et de fixer le menu sur le côté gauche de la page à 140px du haut de cette dernière.
Comme on peut le voir, on a placé le menu à -30px du bord gauche de la page, de façon à masquer la moitié des images utilisées pour le menu. En faisant cela, on s'assure que le menu ne viendra pas recouvrir le contenu de nos pages.
Un peu plus loin dans cet article nous verrons comment afficher le menu avec un effet de slide grâce aux transitions CSS.
Ensuite nous allons créer les items du menu ; nous avons déjà les images qui sont insérées dans le code HTML, il ne nous reste plus qu'à appliquer un style aux boites.

```
li{
        height:50px;
        width:50px;
        background: #7d7e7d; /* Old browsers */
background: -moz-linear-gradient(top, #7d7e7d 0%, #0e0e0e 100%); /* FF3.6+ */
background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,#7d7e7d), color-stop(100%,#0e0e0e)); /* Chrome,Safari4+ */
background: -webkit-linear-gradient(top, #7d7e7d 0%,#0e0e0e 100%); /* Chrome10+,Safari5.1+ */
background: -o-linear-gradient(top, #7d7e7d 0%,#0e0e0e 100%); /* Opera 11.10+ */
background: -ms-linear-gradient(top, #7d7e7d 0%,#0e0e0e 100%); /* IE10+ */
background: linear-gradient(top, #7d7e7d 0%,#0e0e0e 100%); /* W3C */
filter:
progid:DXImageTransform.Microsoft.gradient( startColorstr='#7d7e7d', endColorstr='#0e0e0e',GradientType=0 ); /* IE6-9 */
padding:10px;
position:relative;
border-radius:3px;
-moz-border-radius:3px;
-webkit-border-radius:3px;
}
```

Le point important de ce code CSS est la position : relative; appliquée à nos items. En effet, c'est grâce à cette propriété que nous pourrons par la suite changer la position de nos éléments au survol de la souris.
Maintenant que nous avons créé les items de notre liste, nous pouvons écrire le code CSS qui va nous permettre de faire glisser nos images (effet de slide) au survol de la souris.

```
li:hover {
    left:30px;
    transition: left 0.3s linear;
    -webkit-transition: left 0.3s linear;
    -moz-transition: left 0.3s linear;
}
```

Au survol de la souris, la position des items va changer pour se déplacer de 30px vers la droite et ainsi afficher nos images entières. Le point important dans ce code est la propriété transition qui va nous permettre de faire glisser nos éléments au lieu de seulement changer la position.
La propriété transition vous permet d'ajouter des effets à vos pages sans avoir besoin d'utiliser le JavaScript. Elle a besoin de trois paramètres : le premier est la propriété CSS concernée par l'effet de transition, le deuxième est la durée de l'effet et le dernier la façon dont l'effet s'applique. Dans notre exemple nous utilisons un effet linear ce qui signifie que la transition va s'effectuer avec la même vitesse du début à la fin.

5. Conclusion

Nous avons vu dans ce tutoriel comment créer un menu de navigation toujours visible avec un effet de slide au survol de la souris. Ce menu est très agréable visuellement pour vos utilisateurs. N'hésitez pas à créer vos propres effets grâce à la propriété transition !

6. Liens

Vous pouvez consulter la démonstration pour avoir un aperçu du rendu dans un navigateur : Lien 12.

Retrouvez l'article de Paul Underwood traduit par Sébastien Germez en ligne : Lien 13

Les transitions en CSS3

Avec l'arrivée de CSS3 nous avons vu qu'il était possible de nous passer de JavaScript ou d'utiliser des images pour des présentations originales et spécifiques. Cependant, nous ne sommes pas au bout de nos surprises ! En effet, CSS3 embarque une nouvelle série de propriétés qui permet d'animer des éléments de notre page Web : les transitions.

1. Introduction

Avant de nous plonger tête baissée dans les exemples et le code, il est important de poser les bases.

Qu'est-ce qu'une transition et quand intervient-elle ?

Une transition est l'étape entre deux états. C'est indiquer comment on passe d'un état A à un état B. Par exemple, vous changez la couleur de fond d'un menu quand on le survole ? Et bien la transition intervient au moment du changement, entre le background: red; initial et background: blue; final.
Plus concrètement : si vous disposez d'un bloc avec une couleur de fond rouge et qu'au survol de celui-ci vous voulez qu'il soit bleu, le changement de couleur se fera directement. Avec les transitions, il est possible de changer la couleur progressivement. Voyez plutôt…
Oups ! Avant de continuer… Vous êtes sous Internet Explorer ? Je suis vraiment navré, mais la suite de cet article ne sera qu'une lecture à titre informatif. Ces propriétés ne sont pas encore prises en charge, même dans sa version 9.

2. Fini le blabla, place au code !

2.1. Tout part de là

Je pense que nous sommes au point sur le principe, voyons maintenant comment on le met en œuvre.

Voici les propriétés que nous allons voir dans la suite de cet article :

Nom	Description	Syntaxe
transition-property	les propriétés CSS à transformer	none | [all | <IDENT>] [',' [all | <IDENT>]]*
transition-duration	la durée de la transition	<time> [, <time>]*
transition-delay	le délai du départ de la transition	<time> [, <time>]*
transition-timing-function	le modèle d'interpolation (accélération, décélération…)	[ease | linear | ease-in | ease-out | ease-in-out | step-start | step-end | steps(<integer>[, [start | end]]?) | cubic-bezier(<number>, <number>, <number>, <number>)] [, [ease | linear | ease-in | ease-out | ease-in-out | step-start | step-end | steps(<number>[, [start | end]]?) | cubic-bezier(<number>, <number>, <number>, <number>)]]*

Sans oublier la propriété raccourcie : transition qui regroupe celles sus-citées.

Firefox a besoin d'une version préfixée (-moz-transition-*) et ce depuis la version 12. Idem pour Chrome et Safari (WebKit oblige), préfixe (-webkit-transition-*) depuis la version 20 pour Google et 5.1 pour Apple. Enfin, Opera, pour qui il faut le préfixe -o depuis la version 12.

Pour des raisons de clarté et de lecture, les exemples ci-dessous ne seront pas préfixés. À vous de vous assurer de la compatibilité de ces sources avec votre navigateur.

Entrons dans le vif du sujet avec ceci :

```
<!doctype html>
<head>
    <title>Les transitions ? C'est trop la classe !</title>
    <meta charset="utf-8" />
    <style type="text/css">
    div {
        width: 300px;
        height: 200px;
        line-height: 200px;
        text-align: center;
        background-color: silver;
        border-radius: 20px;
        color: white;
        font-weight: bold;
        text-shadow: 1px 1px 2px black;

        /* Seuls la couleur de fond et les arrondis seront affectés par la transition */
        transition-property: background-color, border-radius;

        /* La transition se fera en une seconde */
        transition-duration: 1s;
    }

    div:hover {
        background-color: #4282b4;
        border-radius: 60px;
        color: yellow;
    }
```

```
    </style>

    <script type='text/javascript'>

    </script>
</head>
<body>
    <header>
    </header>

    <article>
        <div><p>Un beau bloc avec des bords
arrondis</p></div>
    </article>

    <footer>
    </footer>
</body>
</html>
```

Bon, ici, rien d'extraordinaire ; une petite page en HTML 5 avec un <title> « très recherché » et un bloc gris arrondi grâce à la propriété CSS3 border-radius avec du texte ombré par text-shadow.

Côté transition maintenant : vous avez remarqué que dans le :hover de la div je souhaite changer la couleur de fond et du texte, ainsi que le rayon du cercle du border-radius. Jusque-là, rien de nouveau. Sauf que si vous essayez, vous vous rendrez compte que tous ces changements se font progressivement.
Essayez, vous verrez : Lien 14

Hop hop hop ! La couleur du texte elle, elle change instantanément ! C'est quoi ces histoires ?

Allez, on décortique pour que vous compreniez bien l'utilisation de ces propriétés.

Si le changement de la couleur du texte n'est pas progressif, c'est tout simplement parce que la propriété color n'est pas référencée dans cette déclaration :

```
transition-property: background-color, border-
radius;
```

Si j'avais ajouté color dans cette déclaration, alors le changement aurait été gradué. Faites le test et vous en aurez la preuve. Si vous voulez que tout change avec une transition, il faut mettre : transition-property: all. À l'inverse, si vous ne souhaitez rien animer, remplacez all par none.

Ensuite, nous avons transition-duration: 1s. Rien de compliqué ici puisque nous voulons que le changement de couleur de fond et des bordures se fasse en une seconde. Je laisse votre curiosité faire le reste au sujet de cette propriété.

Attention quand même, par défaut, la durée de l'animation est à zéro. Et si c'est à zéro, bah y a pas d'animation ! Et ce n'est pas ce que l'on souhaite.

Voici la liste des propriétés pouvant être animées, donnée par le W3C (Lien 15):

2.2. Laissons le temps au temps

Comme vous le savez, parce que vous venez juste de le lire, il est possible de définir quand la transition va avoir lieu en utilisant transition-delay.

Le même exemple mais avec effet retardé :

```
<!doctype html>
<html lang="fr">
<head>
  <meta charset="utf-8">
  <title>Les transitions ? C'est trop la classe !
</title>
  <style type="text/css">
  div {
    width: 300px;
    height: 200px;
    padding-top: 10px;
    text-align: center;
    background-color: silver;
    border-radius: 20px;
    color: white;
    font-weight: bold;
    text-shadow: 1px 1px 2px black;
    font-size: 30px;

    /* Seuls la couleur de fond et les arrondis
seront affectés par la transition */
    transition-property: background, border-
radius;

          /* La transition se fera en une seconde
*/
          transition-duration: 1s;

    /* On retarde le lancement de la transition
d'une seconde */
    transition-delay: 1000ms;
  }

  div:hover {
    background: #4282b4;
    border-radius: 60px;
    color: yellow;
  }
  </style>
</head>
<body>
  <header></header>
  <section>
    <div><p>Un beau bloc avec des bords
arrondis</p></div>
  </section>
  <footer></footer>
</body>
</html>
```

Ça se passe par ici pour essayer : Lien 16
On se rend compte que l'animation se déclenche 1000 ms (oui j'ai eu le temps de compter, pas vous ?) après le survol de la div.

Cette propriété accepte deux unités de temps pour sa valeur : la seconde notée "s" et la milliseconde "ms".

2.3. C'est quand qu'on arrive ?

Passons maintenant à une des propriétés les plus intéressantes, j'ai nommé: transition-timing-function!
Je vais m'adresser un petit temps aux personnes utilisant jQuery UI. Je pense que vous avez déjà utilisé ou au moins entendu parler de Easing (Lien 17) ? Non ? Boh, ce n'est pas grave, on peut s'en passer maintenant !
Voici lesdites options :

- linear : vitesse de transition constante ;
- ease : vitesse rapide au début et décélère sur la fin. Valeur par défaut ;
- ease-in : accélération constante ;
- ease-out : décélération constante ;
- ease-in-out : vitesse de départ et de fin lente ;
- cubic-bezier(x, x, x, x) : pour les plus matheux d'entre vous, voici la courbe de Bézier. En gros, vous passez quatre paramètres acceptant des valeurs de 0 à 1. Un petit coup de pouce pour rédiger votre courbe de Bézier : Lien 18.

Mis en situation, cela donne :

```
<!doctype html>
<head>
    <title>Les transitions ? C'est trop la classe !</title>
    <meta charset="utf-8" />
    <style type="text/css">
    div {
        float: left;
        width: 150px;
        height: 50px;
        line-height: 50px;
        text-align: center;
        background-color: silver;
        border-radius: 10px;
        color: white;
        font-size: 12px;
        font-weight: bold;
        text-shadow: 1px 1px 2px black;
        margin-bottom: 10px;
        margin-right: 20px;
        transition-property: height, line-height;
        transition-duration: 2s;
    }

    #div1 {
        transition-timing-function: ease;
    }
    #div2 {
        transition-timing-function: ease-in;
    }
    #div3 {
        transition-timing-function: ease-out;
    }
    #div4 {
        transition-timing-function: ease-in-out;
    }
    #div5 {
        transition-timing-function: linear;
    }
    #div6 {
        transition-timing-function: cubic-bezier(0.680, 0.550, 0.265, 1.000);
    }

    button {
        margin-bottom: 10px;
        margin-right: 10px;
    }
    </style>

    <script type='text/javascript'>
    function go() {
        for ( var i = 1;  i < 7; i++ ){
            document.getElementById('div' + i).style.height = "200px";
            document.getElementById('div' + i).style.lineHeight = "200px";
        }
    }
    function dego() {
        for ( var i = 1;  i < 7; i++ ){
            document.getElementById('div' + i).style.height = "50px";
            document.getElementById('div' + i).style.lineHeight = "50px";
        }
    }
    </script>
</head>
<body>
    <header>
    </header>

    <article>
        <p><button onclick="go();">Partez ! </button><button onclick="dego();">Mais pas si vite !</button></p>
        <div id="div1">Fonction : EASE</div>
        <div id="div2">Fonction : EASE-IN</div>
        <div id="div3">Fonction : EASE-OUT</div>
        <div id="div4">Fonction : EASE-IN-OUT</div>
        <div id="div5">Fonction : LINEAR</div>
        <div id="div6">Fonction : CUBIC-BEZIER</div>
    </article>

    <footer>
    </footer>
</body>
</html>
```

3 … 2 … 1 … Partez : Lien 19 !
Alors comme vous pouvez le constater, il y a du JavaScript. Il n'est là que pour lancer les transitions et n'est pas du tout obligatoire.
Vous noterez au passage la puissance de la fonction cubic-bezier. Avec l'aide d'un générateur, les animations seront du plus bel effet (ou avec de la chance en mettant des valeurs aléatoires).

Bref, vous savez faire des transitions ! Ce n'est pas si sorcier et ça apporte un joli plus à votre page Web. Attention toutefois à ne pas tomber dans l'excès.

2.4. Prenons un raccourci

Comme la plupart du temps en CSS, il existe des propriétés raccourcies. Elles combinent plusieurs autres propriétés telles que font, border, background, transition…

Oh bah tiens ! Comme par hasard je parle de transition dans cet article ! Ça tombe bien ça dit donc !?

Reprenons notre exemple :

```
<!doctype html>
<head>
    <title>Les transitions ? C'est trop la classe
!</title>
    <meta charset="utf-8" />
    <style type="text/css">
    div {
        width: 300px;
        height: 200px;
        line-height: 200px;
        text-align: center;
        background-color: silver;
        border-radius: 20px;
        color: white;
        font-weight: bold;
        text-shadow: 1px 1px 2px black;

        transition: border-radius 1s ease-in-out,
background-color 500ms cubic-bezier(0.680, 0.550,
0.265, 1.000) 500ms;
    }

    div:hover {
        background-color: #4282b4;
        border-radius: 60px;
        color: yellow;
    }
    </style>

    <script type='text/javascript'>

    </script>
</head>
<body>
    <header>
    </header>

    <article>
        <div><p>Un beau bloc avec des bords
arrondis</p></div>
    </article>

    <footer>
    </footer>
</body>
</html>
```

C'est par ici que ça se passe pour l'exemple : Lien 20.
On regroupe donc toutes les propriétés que l'on a vues jusqu'à maintenant en respectant la syntaxe donnée en début d'article.

Vous aurez également remarqué que j'ai séparé par une virgule les transitions par propriété à animer.

3. L'exemple en plus

Vous trouverez ici un exemple, basique, de menu avec un hover animé par une transition.

Ainsi qu'un avant-goût du prochain article à venir sur les transformations.

Par ici je vous prie : Lien 21.

4. Conclusion

Vous savez maintenant ce qu'est une transition et comment elle se déclare. Comme je l'ai évoqué et que vous l'avez certainement vu dans mon dernier exemple, nous verrons dans un prochain chapitre l'utilisation des transformations. En espérant que vous avez fait bonne lecture.

Retrouvez l'article de Torgar en ligne : Lien 22

Comprendre le méta tag viewport - Afin d'avoir un « responsive design » pour votre site Web

Dans cet article nous allons apprendre à utiliser le méta tag viewport afin d'adapter le design de votre site web.

1. Traduction

Ce tutoriel est la traduction la plus fidèle possible du tutoriel original de Paul Underwood, Understanding The Viewport Meta Tag : Lien 23.

2. Introduction

Quand vous travaillez sur un nouveau design de site Web, une des choses auxquelles il faut penser est le « responsive design ». Est-ce que le site que vous allez développer a besoin d'un « responsive design » ?
Tous les sites Web aujourd'hui devraient prendre en considération l'aspect « responsive design ». Il y a encore certains développeurs qui pensent qu'un visiteur sur mobile peut toujours zoomer pour rendre une page lisible, mais le sentiment général est que de nos jours, chaque développeur devrait rendre son site facile d'accès et d'utilisation.

3. Qu'est-ce que le « responsive design » ?

Le « responsive design » c'est lorsque le design de votre site s'adapte à la résolution de l'écran de votre visiteur. Ce concept se met en œuvre avec l'utilisation des « media queries » CSS3. Ces derniers vous permettent de modifier la largeur de vos éléments HTML en fonction de paramétrages que vous définissez dans vos feuilles de style. En ajoutant simplement width: 100 % en CSS sur vos éléments HTML peut suffire à les rendre « responsive » pour des appareils mobiles.
Les différents paramétrages définissent chacun une fourchette de largeur qui utilisera un style particulier. C'est typiquement des largeurs d'affichage différentes en fonction de l'appareil utilisé pour visiter le site : un smartphone, une tablette, un PC de bureau, etc.
Vous pouvez définir ces paramétrages directement avec une largeur en pixels ou bien avec un ratio minimum en pixels.
Le code suivant peut vous aider à démarrer avec les « media queries ».

```
/* Smartphones (portrait and landscape)
----------- */
@media only screen
and (min-width : 320px)
and (max-width : 480px) {
/* Styles */
}
/* Smartphones (landscape) ----------- */
@media only screen
and (min-width : 321px) {
/* Styles */
}
/* Smartphones (portrait) ----------- */
@media only screen
and (max-width : 320px) {
/* Styles */
}
/* iPads (portrait and landscape) ----------- */
@media only screen
and (min-width : 768px)
and (max-width : 1024px) {
/* Styles */
}
/* iPads (landscape) ----------- */
@media only screen
and (min-width : 768px)
and (max-width : 1024px)
and (orientation : landscape) {
/* Styles */
}
/* iPads (portrait) ----------- */
@media only screen
and (min-width : 768px)
and (max-width : 1024px)
and (orientation : portrait) {
/* Styles */
}
/* Desktops and laptops ----------- */
@media only screen
and (min-width : 1224px) {
/* Styles */
}
/* Large screens ----------- */
@media only screen
and (min-width : 1824px) {
/* Styles */
}
/* iPhone 4 ----------- */
@media
only screen and (-webkit-min-device-pixel-ratio :
1.5),
only screen and (min-device-pixel-ratio : 1.5) {
/* Styles */
}
```

4. Définir le « viewport »

Le méta tag « viewport » dicte au navigateur quel comportement il doit adopter pour afficher une page ; vous pouvez spécifier la largeur qu'adoptera le « viewport ».
Le « viewport » correspond à la partie de la page affichée à l'écran. Par exemple si vous visitez un site Web depuis un appareil mobile, et qu'à l'écran s'affiche le coin haut gauche de la page (comme s'il y avait un zoom) c'est que le « viewport » a été défini à une certaine largeur. En revanche, si la page entière du site s'affiche et qu'elle occupe tout votre écran, c'est que le « viewport » a été défini pour afficher toute la page.

Ci-dessous un exemple d'utilisation du méta tag « viewport ».

```
<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
```

Cette ligne définit la largeur du « viewport » pour être la même que la taille de l'écran de l'appareil utilisé pour afficher le site. L'échelle d'affichage du site sera 100 % et l'échelle maximum également de 100 %.
Chaque navigateur mobile dispose de son propre « viewport » par défaut, la plupart de ces navigateurs utilisant un « viewport » par défaut de 980px ce qui signifie que la page sera affichée dans une section d'affichage de 980 pixels et qu'un zoom arrière sera effectué si nécessaire pour que la page s'affiche en plein écran.
Plus l'écran de votre appareil sera petit plus le zoom arrière sera important. Si vous comparez l'affichage d'une même page Web sur un iPhone et sur un iPad, le zoom sera plus important sur l'iPhone que sur l'iPad.
Changer la valeur du « viewport » vous permet de personnaliser la façon dont le navigateur va afficher votre site.

5. La largeur du « viewport »

La largeur que vous définissez dans votre « viewport » revient à dire que c'est la largeur qui permet l'affichage le plus optimisé de votre site. Si vous avez développé une page optimisée pour un affichage sur iPhone, alors la largeur de votre « viewport » sera de 320 pixels.

```
<meta name="viewport" content="width=320">
```

Cependant, cette façon de procéder ne permet pas de faire du « responsive design ». En effet, si vous visualisez cette page sur une tablette, vous aurez un zoom arrière sur la page pour qu'elle s'affiche dans une section de 320 pixels seulement, laissant le reste de l'écran vide.
La meilleure solution pour que votre site ait un « responsive design » est de définir comme largeur de « viewport » la largeur du matériel utilisé pour visualiser la page Web.

```
<meta name="viewport" content="width=device-width">
```

6. Les échelles du « viewport »

Sur les appareils mobiles vous pouvez zoomer (en avant ou en arrière) grâce à un pincement de doigts, mais si vous définissez la largeur du « viewport » pour être celle de l'appareil utilisé pour visualiser la page, vous n'aurez pas besoin d'utiliser de zoom. Pour vous assurer que votre page s'affichera sans zoom, vous pouvez utiliser la propriété inital-scale pour définir le zoom par défaut.
En mettant l'échelle à 1, on s'assure qu'aucun zoom ne sera fait à l'affichage de la page.

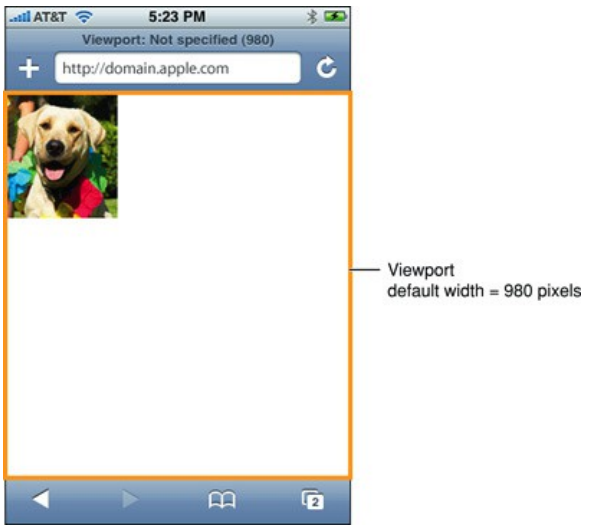
```
<meta name="viewport" content="initial-scale=1">
```

Si vous souhaitez empêcher tout zoom de la part de l'utilisateur, vous pouvez définir la propriété maximum-scale à 1 afin d'empêcher tout redimensionnement.

```
<meta name="viewport" content="maximum-scale=1">
```

7. Exemples de « viewport »

Les images suivantes montrent l'affichage de la même page avec deux « viewport » différents. La première image affiche la page dans un « viewport » de 980 pixels et la seconde dans un « viewport » de 320 pixels.



8. Conclusion

Nous avons vu dans ce tutoriel comment utiliser le méta tag viewport afin d'avoir un « responsive design » sur notre site Web. Ce méta tag permet d'avoir un affichage de page dynamique en fonction de l'appareil utilisé pour afficher cette dernière (smartphone, tablette, laptop, etc.).

Retrouvez l'article de Paul Underwood traduit par Sébastien Germez en ligne : Lien 24

Comment ajouter des liens vers des numéros de téléphone - Grâce à une nouveauté HTML 5

Dans ce tutoriel nous allons voir comment faire en sorte que les numéros de téléphone présents dans vos pages web deviennent cliquables de la même façon que les adresses e-mail le sont.

1. Traduction

Ce tutoriel est la traduction la plus fidèle possible du tutoriel original de Paul Underwood, Add Telephone Number Links With HTML5 : Lien 25.

2. Introduction

Jusqu'à aujourd'hui, les numéros de téléphone présents dans les pages web étaient ajoutés en tant que texte simple (statique) et n'offraient donc aucune interaction pour les visiteurs. Le nombre de possesseurs de smartphone et donc d'utilisateurs du web via un téléphone étant en augmentation constante, il est temps de changer ces numéros de téléphone pour des zones cliquables afin de pouvoir directement appeler ledit numéro par un simple clic.
En effet à l'heure actuelle, on est obligé de copier le numéro de téléphone qui nous intéresse dans la page, puis de le recopier dans l'interface d'appel du téléphone.
Avec cette nouvelle fonctionnalité, on va pouvoir créer des liens à partir de numéros de téléphone de la même façon que l'on crée des liens à partir d'adresses e-mail.

3. « href=" tel :" » semblable à « href=" mailto :" »

Pour permettre aux visiteurs de directement envoyer un e-mail en cliquant sur une adresse depuis une page web, on utilise un lien classique dans lequel on place mailto : dans l'attribut href.

```
<a href="mailto:example@example.com">Example Email</a>
```

Après avoir cliqué sur ce lien, le client de messagerie par défaut s'ouvre avec un nouvel e-mail dont le champ destinataire est rempli à partir de l'adresse spécifiée dans l'attribut href du lien.
Suivant ce modèle, pour créer un lien cliquable vers un numéro de téléphone, il faut utiliser tel: dans l'attribut href du lien en précisant le numéro à appeler.

```
Book now, call <a href="tel:01234567890">01234 567 890</a>
```

4. Skype

Il est également possible de créer un lien cliquable qui utilisera Skype pour appeler le numéro de téléphone renseigné dans le lien (à partir d'un ordinateur de bureau par exemple).
Pour ce faire, il suffit de remplacer la valeur tel par callto dans l'attribut href du lien.

```
Book now, call <a href="callto:01234567890">01234 567 890</a>
```

5. Conclusion

Nous avons vu dans ce tutoriel comment rendre interactifs les numéros de téléphone présents dans les pages web. La croissance des visiteurs utilisant des smartphones étant en constante augmentation, cette nouveauté disponible avec HTML 5 vous permet de rendre vos pages web plus attrayantes pour ces utilisateurs.

Retrouvez l'article de Paul Underwood traduit par Sébastien Germez en ligne : Lien 26

Utiliser l'API PageVisibility

Cet article est la traduction de Using the PageVisibility API (Lien 27), par Joe Marini publié sur le site HTML5 Rocks (Lien 28).

1. Introduction

En tant que développeurs Web, nous sommes toujours curieux des nouvelles techniques permettant d'obtenir des pages Web plus riches et interactives.
Des graphiques 3D avec WebGL ? Absolument ! Des capacités audio avancées avec WebAudio ? Je prends ! Des applications participatives en temps réel utilisant la webcam et le micro ? Je signe où ?
Moins spectaculaires mais tout aussi importantes, des technologies existent permettant de créer des applications plus efficaces et améliorant l'expérience de l'utilisateur. L'*API Page Visibility* (Lien 29) en fait partie. Cette API a un rôle simple, mais néanmoins important : elle permet à votre application de savoir si une page est visible par l'utilisateur. Grâce à cette information, vous pourrez créer des pages Web qui se comporteront différemment selon qu'elles sont visibles ou non. Prenons quelques exemples :

- une page récupérant des informations depuis le serveur peut ralentir son cycle de mise à jour lorsqu'elle n'est pas visible ;
- une page affichant une galerie d'images ou du contenu audio/vidéo peut se mettre en pause tant que l'utilisateur ne revient pas sur la page ;
- il se peut qu'une application n'affiche de notifications que lorsque la page n'est pas visible.

Au premier abord, cette API peut ne pas sembler très utile pour l'utilisateur, mais si l'on considère l'augmentation de la navigation depuis un appareil mobile, toute technique permettant d'économiser la batterie devient rapidement importante. En utilisant l'*API PageVisibility*, votre site pourra aider l'appareil à consommer moins d'énergie et donc augmenter son autonomie.
La spécification de l'API, qui n'en est, au moment d'écrire cet article (N.D.T. : 25/10/2012), qu'à l'état de recommandation, propose des propriétés pour connaître l'état de visibilité du document, ainsi que des événements indiquant le changement d'état. Dans cet article, je traiterai des bases de l'API et montrerai des exemples concrets d'utilisation.

2. Propriétés de visibilité du document

La version courante des spécifications de l'API PageVisibility (Lien 29) définit deux propriétés pour le document : la valeur booléenne hidden et l'énumération visibilityState. La propriété visibilityState peut avoir quatre valeurs : hidden, visible, prerender et unloaded.

Actuellement, ces propriétés ne sont disponibles qu'à l'aide de préfixes vendeurs. Vous devrez donc utiliser les versions préfixées (comme webkitHidden ou webkitVisibilityState) tant que la spécification n'aura pas le statut de recommandation et que les navigateurs n'implémenteront pas les versions sans préfixe.

Comme vous pouvez vous y attendre, l'attribut hidden renvoie true si le document n'est pas visible. Typiquement, cela signifie que le document est réduit, dans un onglet non actif, etc. L'attribut vaut false si au moins une portion du document est partiellement visible sur au moins un écran. De plus, afin de s'adapter aux outils d'accessibilité, l'attribut hidden peut valoir false si un outil comme une loupe masque le document en en montrant une vue.

2.1. Gérer les préfixes vendeurs

Afin de rester concentré sur le code et non sur tous les préfixes vendeurs spécifiques, je vais créer des fonctions utilitaires pour gérer les différentes notations.

```
function getHiddenProp(){
    var prefixes = ['webkit','moz','ms','o'];

    // si 'hidden' est supporté nativement, le
retourner
    if ('hidden' in document) return 'hidden';

    // sinon, boucler sur les préfixes connus
pour trouver le bon
    for (var i = 0; i < prefixes.length; i++){
        if ((prefixes[i] + 'Hidden') in document)
            return prefixes[i] + 'Hidden';
    }

    // sinon, la fonctionnalité n'est pas
supportée
    return null;
}
```

2.2. Exemple de propriétés du document

Nous pouvons maintenant écrire une fonction compatible pour tous les navigateurs, isHidden(), pour savoir si le document est visible.

```
function isHidden() {
    var prop = getHiddenProp();
    if (!prop) return false;

    return document[prop];
}
```

Pour connaitre de façon plus précise l'état de visibilité du document, on utilisera plutôt la propriété visibilityState. Cette propriété retourne l'une des quatre valeurs suivantes :

- hidden : le document est complètement caché ;

- visible : au moins une partie du document est visible sur au moins un écran ;
- prerender : le document est chargé hors écran et n'est pas visible (cette valeur étant optionnelle, tous les navigateurs ne l'implémentent pas nécessairement : Lien 30) ;
- unloaded : le document n'est pas chargé (cette valeur étant optionnelle, tous les navigateurs ne l'implémentent pas nécessairement : Lien 30).

3. L'événement visibilityChange

En complément des propriétés de visibilité, il existe un événement visibilityChange déclenché chaque fois que l'état de la visibilité du document change. Vous pouvez donc utiliser un gestionnaire d'événements directement sur l'objet document :

```
// utiliser les noms de propriété pour trouver le nom de l'événement préfixé
var visProp = getHiddenProp();
if (visProp) {
  var evtname = visProp.replace(/[H|h]idden/,'') + 'visibilitychange';
  document.addEventListener(evtname, visChange);
}

function visChange() {
   var txtFld = document.getElementById('visChangeText');

   if (txtFld) {
      if (isHidden())
         txtFld.value += "Onglet non visible !\n";
      else
         txtFld.value += "Onglet visible !\n";
   }
}
```

Si le message « L'API PageVisibility n'est pas supportée ! » est affiché, c'est que votre navigateur ne supporte pas cette API.

4. Exemples pratiques

4.1. Démarrer et mettre en pause une vidéo

Cet exemple montre comment utiliser l'API PageVisibility pour lancer et mettre en pause une vidéo.
Lancez la vidéo (si elle n'est pas supportée, testez avec Chrome), puis changez d'onglet. Vous pourrez alors constater qu'elle se met automatiquement en pause, puis reprend la lecture lorsque vous revenez sur l'onglet.

Ceci est réalisé en utilisant un gestionnaire pour l'événement visibilityChange et en modifiant l'état de lecture de la vidéo.

```
window.addEventListener("load", function vidDemo() {
   sessionStorage.initialPlay = false;
   var vidElem = document.getElementById("video-demo");

   var visProp = getHiddenProp();
   if (visProp) {
      vidElem.addEventListener("play", function() {
         sessionStorage.initialPlay = true;
      });

      var evtName = visProp.replace(/[H|h]idden/,'') + 'visibilitychange';
      document.addEventListener(evtName, startStopVideo);
   }

   function startStopVideo() {
      if (document[visProp]) {
         vidElem.pause();
      }
      else if (vidElem.paused && sessionStorage.initialPlay == "true") {
         vidElem.play();
      }
   }
});
```

4.2. Afficher une notification lorsque l'onglet est masqué

L'API Notifications (Lien 31) permet d'afficher des bulles d'informations au niveau du navigateur pour attirer l'attention de l'utilisateur. Cependant, lorsque l'utilisateur se trouve déjà sur la page, attirer son attention n'est pas nécessairement utile. Grâce à l'API PageVisibility, nous allons pouvoir afficher les notifications uniquement lorsque l'onglet n'est pas actif.
Vous devez d'abord activer les notifications (en cliquant sur le bouton ci-dessous avec un navigateur compatible, par exemple Chrome). Cliquez ensuite sur « Envoyer une notification ! » et changez d'onglet. Après cinq secondes, le code va vérifier l'état de l'onglet et afficher soit une notification, soit une boîte d'alerte classique.

Voici le code correspondant :

```
window.addEventListener("load", function notifyDemo() {
   var propName = "";
   var oNotify=null;
   var visProp = getHiddenProp();

   document.getElementById("notify-demo").addEventListener("click", function() {
      oNotify = null;
      if (window.webkitNotifications) {
         setTimeout(showNotification, 5000);

         if (window.webkitNotifications.checkPermission() == 0) { // 0 = PERMISSION_ALLOWED
            oNotify = window.webkitNotifications.createNotification('', 'Notification', 'Vous avez reçu une notification !');
         }
      }
   });

   document.getElementById("notify-enable").addEventListener("click", function() {

window.webkitNotifications.requestPermission();
   });
```

```
   function showNotification() {
      if (document[visProp] &&
window.webkitNotifications && oNotify) {
         oNotify.show();
      }
      else {
         alert("Vous avez reçu une
notification !");
      }
   }
});
```

4.3. Gérer Google Analytics si la page est préchargée

Certains navigateurs (comme Chrome), permettent de précharger des pages (voir plus d'informations ici : Lien 32). Ce procédé implique de charger toutes les ressources nécessaires au rendu de la page, y compris les scripts. De nombreux sites utilisent *Google Analytics* pour avoir des statistiques sur la fréquentation du site. Cependant, ces données peuvent être faussées si certaines pages sont préchargées mais jamais réellement vues.
Pour pallier ce problème, vous pouvez utiliser l'API PageVisibility afin de savoir si la page est en mode de préchargement et, dans ce cas, ne pas charger le script. La page contient normalement une portion de code similaire à celui-ci :

```
var _gaq = _gaq || [];
_gaq.push(['_setAccount', 'UA-XXXXX-X']);
_gaq.push(['_trackPageview']);
```

Dans ce cas, l'événement _trackPageview est toujours déclenché, même si la page n'est pas affichée. Un script permettant de détecter le préchargement ressemblerait à celui-ci :

```
var bHavePV = getHiddenProp();
var bInitialShow = false;
var _gaq = _gaq || [];
_gaq.push(['_setAccount', 'UA-XXXXX-X']);

if (bHavePV) {
   document.addEventListener("visibilityChange",
handleVisEvt);
}
else {
   _gaq.push(['_trackPageview']); // si l'API
n'est pas reconnue, charger normalement le script
}

function handleVisEvt() {
   if (document.visibilityState == "prerender") {
      _trackEvent("pagedata", "prerender"); //
garder une trace du préchargement
   }
   if (document.visibilityState == "visible" && !
bInitialShow) {
      bInitialShow = true; // ne charger le code
qu'une fois
      _gaq.push(['_trackPageview']);
   }
}
```

5. Conclusion et références

5.1. Conclusion

Créer une bonne application Web ne se limite pas à des effets spectaculaires ou des interactions poussées avec l'utilisateur. Il est aussi nécessaire de prendre en considération et de préserver les ressources de l'appareil de l'utilisateur. De ce point de vue, l'API PageVisibility est d'un grand intérêt. Pour aller plus loin sur ce sujet, vous pouvez consulter les autres articles liés aux performances de **HTML5 Rocks** : Lien 33.

5.2. Références

- Spécifications de l'API PageVisibility (W3C) : Lien 29.
- Groupe de travail sur les performances Web : Lien 34.
- Exemple de PageVisibility pour IE10 : Lien 35.
- Visibility.js : solution de contournement de PageVisibility : Lien 36.
- IsVis : autre solution de remplacement : Lien 37.

Retrouvez l'article de Joe Marini traduit par Didier Mouronval en ligne : Lien 38

L'API JavaScript Full Screen

Dans ce tutoriel nous allons voir comment utiliser l'API JavaScript full screen sur des éléments HTML.

1. Traduction

Ce tutoriel est la traduction la plus fidèle possible du tutoriel original de Paul Underwood, Javascript FullScreen API : Lien 39.

2. Introduction

L'API full screen (mode plein écran) est un moyen facile d'obtenir le contenu Web complet que vous voulez afficher en mode plein écran. Il est très similaire à la touche F11 de votre clavier, mais laisse le choix aux développeurs. C'est une fonctionnalité intéressante à utiliser par exemple avec des diaporamas, vous pouvez renverser le navigateur en mode plein écran via le clic sur une image.

La meilleure chose à propos de cette fonctionnalité est que l'élément ne doit pas être une page entière, vous pouvez mettre n'importe quel élément HTML en plein écran. Cela signifie que plutôt que de toujours mettre la page entière en mode plein écran, vous n'avez qu'à assigner l'API full screen à une image et au moment du clic sur l'image, vous n'afficherez que l'image en mode plein écran.

3. Navigateurs supportés

Malheureusement, tous les navigateurs ne prennent pas en charge cette fonctionnalité donc vous avez besoin de vous assurer que le navigateur de vos visiteurs prend en charge cette fonctionnalité avant d'essayer de l'utiliser.
Si le navigateur qu'ils utilisent supporte l'API, alors vous

pouvez l'utiliser. Si le navigateur ne la supporte pas, alors vous pouvez être sûr que vous allez devoir faire autre chose que d'utiliser le mode plein écran.
Selon le site Can I Use (Lien 40), sont actuellement compatibles les navigateurs suivants :

- Chrome 15+ ;
- Firefox 10+ ;
- Safari 5.1+ ;
- Opera 12.50+.

Safari 5.1 ne supporte pas l'utilisation du clavier en mode plein écran.

3.1. Vérifier que le navigateur supporte l'API

Pour vérifier si un navigateur prend en charge cette fonctionnalité, vous pouvez utiliser l'extrait de code suivant :

```
var docElm = document.documentElement;
if (docElm.requestFullscreen) {
    docElm.requestFullscreen();
}
else if (docElm.mozRequestFullScreen) {
    docElm.mozRequestFullScreen();
}
else if (docElm.webkitRequestFullScreen) {
    docElm.webkitRequestFullScreen();
}
```

Ce test utilise la méthode requestFullscreen. Si elle retourne true, vous pouvez utiliser l'API.
Firefox et Chrome ont leurs propres méthodes pour cela et sont préfixées par moz et webkit, c'est-à-dire mozRequestFullScreen et webkitRequestFullScreen. Une fois ces méthodes détectées, vous pouvez utiliser ensuite la méthode du mode plein écran.

3.2. Quitter le mode plein écran

Pour quitter le mode plein écran, il y a également une méthode. Voici un extrait de code que vous pouvez associer à un bouton sur votre page pour annuler le mode plein écran et revenir à l'affichage normal :

```
if (document.exitFullscreen) {
    document.exitFullscreen();
}
else if (document.mozCancelFullScreen) {
    document.mozCancelFullScreen();
}
else if (document.webkitCancelFullScreen) {
    document.webkitCancelFullScreen();
}
```

3.3. Personnalisation

C'est possible grâce à cette fonctionnalité CSS livrée avec des nouvelles pseudoclasses (Lien 41) permettant de modifier le style uniquement en mode plein écran :

```
html:-moz-full-screen {
    background: red;
}
html:-webkit-full-screen {
    background: red;
}
html:fullscreen {
    background: red;
}
```

4. Le projet Screenfull.js sur GitHub

J'ai récemment découvert un bon plugin JavaScript sur GitHub qui agit comme une couche pour l'API du mode plein écran du HTML5. L'avantage d'utiliser ce plugin au lieu du JavaScript brut est qu'il gère tous les problèmes de compatibilité des navigateurs que vous obtenez lorsque vous travaillez avec le JavaScript brut.
Un attribut statique est visible à la fois depuis la classe et depuis n'importe quelle instance de la classe.
Pour utiliser screenfull.js, c'est très facile. Tout ce que vous devez faire est d'utiliser l'extrait de code suivant :

```
if ( screenfull ) {
    screenfull.request();
}
```

Pour télécharger screenfull.js, vous devez aller sur la page GitHub du projet : Lien 42.

5. Exemples d'utilisation

Pour utiliser screenfull.js, c'est très simple. Voici quelques exemples d'utilisation.
Pour vérifier si le mode plein écran est pris en charge, utilisez l'extrait de code suivant :

```
<script>
$(function() {
if(screenfull)
{
    $('#supported').text('Oui');
} else {
    $('#supported').text('Non');
}
});
</script>
Votre navigateur supporte-t-il l'API full
screen ? <span id="supported"></span>
```

Lorsque vous savez que le navigateur prend en charge l'API, vous pouvez activer le mode plein écran sur un événement click d'un bouton tout simplement en utilisant ce code :

```
var target = $('#target')[0]; // Récupère
l'élément du DOM d'après la collection d'objets
jQuery
$('#button').click(function() {
    if ( screenfull ) {
        screenfull.request( target );
    }
});
```

Vous pouvez également joindre cela à toutes les images de la page à l'aide de cet extrait de code :

```
$('img').click(function() {
     if ( screenfull ) {
     screenfull.toggle( this );
     }
});
```

Regardez la démonstration pour voir des exemples d'utilisation de screenfull.js : Lien 43.

6. Conclusion

Nous avons vu dans ce tutoriel comment utiliser l'API JavaScript full screen sur des éléments HTML.

Retrouvez l'article de Paul Underwood traduit par Sébastien Germez et vermine en ligne : Lien 44

Necessitas, le port Android de Qt, intégré au Qt Project

Il était déjà annoncé que Digia envisageait de supporter les plateformes mobiles de mieux en mieux avec Qt, avec pour objectif l'intégration d'Android et iOS dans les plateformes supportées en *tier 1* courant 2013 (c'est-à-dire comme plateformes principales). Une des pistes était l'intégration du code de Necessitas, le port lancé par BogDan Vatra pour Android : la bonne nouvelle du jour est que les deux parties se sont mises d'accord pour que cela arrive ! Le port Android de Qt 5 sera basé sur le projet Necessitas, BogDan souhaitant en devenir mainteneur (selon l'organisation du Qt Project : Lien 45).

Necessitas est développé sous l'égide du projet KDE, qui a fourni l'infrastructure nécessaire pour le développement et la distribution ; il est actuellement utilisé par un grand nombre de développeurs, qui ont publié des applications l'utilisant sur des périphériques Android.

Ce projet a été lancé grâce à QPA, la plateforme d'abstraction des plateformes de Qt, apparue avec la version 4.8 pour faciliter le port de Qt vers d'autres plateformes. Faisant ses débuts fin 2010, la première bêta a été rendue disponible en février 2011, le port ne cessant depuis lors de supporter de plus en plus de fonctionnalités. Actuellement, le port n'est toujours pas considéré comme fini, mais cela devrait arriver dans les mois à venir, ces efforts pouvant être repris en très grande partie pour le port Qt 5.

Le projet Necessitas ne contient pas qu'un port de Qt : on compte également le Necessitas SDK, qui ajoute à Qt Creator le support d'Android, ainsi que Ministro, qui télécharge les dépendances (Qt) des applications déployées.

Pour ceux qui sont à Berlin cette semaine, quelques démos devraient avoir lieu aux DevDays (reportage en live : Lien 46).

Commentez la news de Thibaut Cuvelier en ligne : Lien 47

En direct des Qt DevDays 2012

Deuxième journée aux Qt Dev Days 2012. La matinée était réservée aux keynotes à propos de Qt 5, l'occasion d'en apprendre plus sur le présent et le futur de Qt 5.

En bref, les objectifs autour du projet Qt sont :

- être numéro 1 dans la gestion de plateformes (Jolla, Blackberry 10, Android, iOS, les plateformes embarquées et bien sûr, les plateformes de bureau) ;
- être numéro 1 dans l'expérience développeur (avec l'intégration de nouveaux outils, une meilleure documentation…) ;
- être numéro 1 dans l'expérience utilisateur (prototypage rapide, performance, fluidité, fonctionnalités…) ;
- garder le modèle de double licences (Open Source/commerciale) ;
- renforcer l'écosystème autour de Qt ;
- continuer avec l'architecture ouverte du Qt Project.

Ensuite, Lars Knoll, mainteneur en chef, est venu nous présenter plus en détail Qt 5, dont la seconde bêta est maintenant disponible (Lien 48).
Tout d'abord, Qt 5 sera normalement disponible en décembre. Une fonctionnalité supplémentaire sera ajoutée tous les six mois. Ainsi, Qt 5.1 verra le jour durant le printemps 2013 et Qt 5.2 durant la fin de l'année 2013.
Actuellement, le support de BlackBerry 10 devrait arriver durant le printemps 2013 et le support de iOS et Android dans Qt 5.2.

Lars a aussi annoncé que le projet Qt était une réussite avec 25 000 commits acceptés venant de 450 commiteurs rien qu'en une année.

Ensuite, la keynote est revenue sur Qt 5.

Comme cela avait déjà été annoncé l'année dernière, le code Qt 4 est compatible avec Qt 5 et cela a été confirmé. Qt Creator 2.6 est capable de compiler un projet avec Qt 4 et avec Qt 5.

Qt 5 aura une architecture modulaire permettant d'utiliser et d'embarquer uniquement ce qui est nécessaire. Certains modules de Qt 4 ont été dépréciés (mais restent présents dans Qt 5) :

- Qt OpenGL ;
- Qt XML ;
- Qt SVG ;
- Qt Script et les outils de scripting.

Finalement, Lars Knoll a revu les différentes améliorations de Qt 5 :

- nouvelle abstraction pour le code spécifique aux plateformes (permet de 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 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.

Commentez la news d'Alexandre Laurent en ligne : Lien 49

Sortie de Qt 5 Bêta 2

Juste pour les Qt DevDays 2012 à Berlin (reportage en live : Lien 49), Qt 5.0 Beta 2 est de sortie, comme prévu. La sortie de la version finale est toujours prévue pour cette fin d'année 2012.

D'ailleurs, tout commence à se mettre au parfum Qt 5 : Qt Creator 2.6, sorti il y a peu, peut être basé sur Qt 5 pour compiler avec Qt 5, formant une version complète de l'environnement Qt très utile pour tester cette nouvelle version majeure.

Cela montre aussi le niveau de compatibilité entre Qt 5 et Qt 4 : le même code source est utilisé pour compiler Qt Creator, peu importe la version de Qt. Qt 4.8 et Qt 5.0 garantissent presque une garantie de compatibilité des sources.

Cette préversion montre aussi une liste des modules et paquets qui feront partie de la version 5.0 finale, ainsi que les distributions entre modules essentiels et add-ons. Mêmes si ces listes paraissent longues (signe d'un grand nombre de fonctionnalités), certains modules – comme Qt 3D – n'en font pas partie, bien qu'ils seront disponibles du Qt Project et utilisables avec Qt 5.0.
Qt Essentials :
Qt Core
Qt Network
Qt Gui
Qt Sql
Qt Testlib
Qt Widgets
Qt Qml
Qt Quick
Qt Multimedia
Qt WebKit
Qt WebKit Widgets
Qt Add-ons :
Qt Xml
Qt XmlPatterns
Qt Svg
Qt Concurrent
Qt Printsupport
Qt Dbus
Qt OpenGL
Qt ActiveQt
Qt Graphical Effects
Qt Script
Qt Declarative
Qt Image Formats

Outils :
Qt Assistant
Qt Linguist
Qt Designer
Qt Translations
Qt Creator
Qt Documentation

En tant que préversion, un certain nombre de problèmes est déjà connu ; notamment, la documentation n'a pas reçu sa dose d'attention, de grands changements sont à prévoir à ce niveau avant la version finale. La liste des problèmes connus : Lien 50.

En tant que projet communautaire, il est important que tous participent à l'élaboration de cette nouvelle version : essayez Qt 5 sur vos projets, donnez votre avis au Qt Project pour résoudre autant de problèmes potentiels que possible avant la version finale.

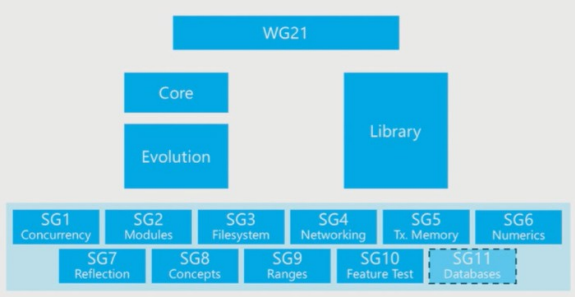
Télécharger Qt 5 Beta 2 : Lien 51

Commentez la news de Thibaut Cuvelier en ligne : Lien 52

Le futur du C++

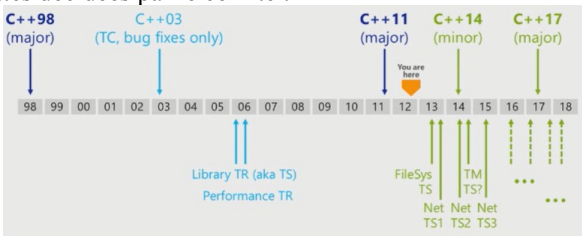
Hier, Herb Sutter a dévoilé plusieurs annonces importantes pour l'avenir du C++.
Notamment les dates des prochains standards, qui ont finalement été décidées.

Il a tout d'abord présenté de façon simple l'organisation du processus de standardisation :



Chaque SG est un groupe de travail qui peut comprendre jusqu'à quarante personnes. Ils étudient les besoins spécifiques à leur domaine afin de proposer la conception d'une fonctionnalité majeure. La conception est ensuite améliorée et mise au point par les groupes Evolution et Library : Evolution est responsable de l'évolution du langage tandis que Library traitera de l'impact de cette fonctionnalité dans la STL. Les groupes Evolution et Library peuvent également proposer des fonctionnalités mineures.
Puis chaque fonctionnalité passe par le Core qui en fera un « draft » détaillé capable d'intégrer le standard. Enfin, ce « draft » passera au vote général du WG21 – qui est le nom du comité international qui traite du C++ - , afin de décider de son intégration au prochain standard, ce qui en fait un traité international.

Cette introduction permet de présenter le calendrier des dates décidées par le comité :



On note bien évidemment deux dates importantes qui correspondent à deux nouveaux standards : C++17 qui sera un nouveau standard majeur, autrement dit de même importance que la norme C++11 ; mais également C++14 qui sera un standard mineur, c'est-à-dire qu'il ne comportera que des ajouts mineurs, mais aussi toutes les fonctionnalités majeures votées à cette date.
Ces fonctionnalités majeures feront l'objet de TS – Technical Specification est le nouveau nom donné aux TR alias Technical Report – indépendants, ce qui permettra de profiter des ajouts majeurs sans attendre 2017.
Le SG4 a d'ores et déjà annoncé qu'il publierait un TS Network tous les ans, mais le TS le plus proche concerne FileSystem qui est la 3e révision de cet ajout tiré de boost.
Le TS sur la mémoire transactionnelle est plus flou dans les dates puisque le SG5 doit pour cela travailler de concert avec le consortium industriel.

Mais ce n'est pas tout, les annonces ne sont pas terminées. La suivante concerne l'ouverture officielle de isocpp.org, qui sera le site officiel du comité de standardisation. On pourra y trouver des informations sur l'avancement de la norme, des annonces importantes concernant le C++, et beaucoup d'autres choses. Le but de ce site est de promouvoir un style de programmation C++ moderne ainsi que l'expansion des bibliothèques C++ portables.

La dernière annonce concerne la création d'une association indépendante, la Standard C++ Foundation – à noter qu'elle est financée par 17 grandes entreprises à ce jour ayant un intérêt commun à la standardisation du C++. Son but est de promouvoir la compréhension et l'utilisation du C++ standard sur tous les compilateurs et toutes les plateformes – les entreprises qui financent ne contrôlent pas l'association. Les six directeurs de cette association sont C. Carruth, B. Stroustrup, B. Dawes, H. Sutter, S. Du Toit et M. Wong.

Toutes ces annonces promettent un bel avenir pour le C++, l'utilisation et la propagation de ces normes internationales ne tiennent qu'à vous tous, développeurs C++.

Comment suivez-vous l'actualité du C++ ?
Connaissiez-vous le fonctionnement du comité de standardisation du C++ ?
Utilisez-vous le dernier standard (C++11) en production ? À titre personnel ?
Comment vous formez-vous à l'utilisation de la dernière norme (C++11) ?

Voir également Microsoft renforce le support de C++ 11 dans Visual Studio 2012 : Lien 53

Commentez la news de Germino Legrand en ligne : Lien 54.

Le C++ expressif n° 1 : introduction

Dissimulé dans C++ se cache un autre langage – d'innombrables autres langages, en fait – tous sont meilleurs que le C++ pour résoudre certains types de problèmes. Ces domain-specific languages (abrégé DSL) sont par exemple des langages pour l'algèbre linéaire ou des langages de requêtes, ils ne peuvent faire qu'une seule chose, mais ils le font bien. On peut créer et utiliser ces langages directement dans le C++, en utilisant la puissance et la flexibilité du C++ pour remplacer les parties communes du langage par les parties spécifiques au domaine que nous utilisons. Dans cette série d'articles, nous allons regarder de près les domain-specific languages, dans quels domaines sont-ils utiles et comment peut-on facilement les implémenter en C++ avec l'aide de Boost.Proto.

1. Introduction

Dissimulé dans C++ se cache un autre langage – d'innombrables autres langages, en fait – tous sont meilleurs que le C++ pour résoudre certains types de problèmes. Ces *domain-specific languages* (abrégé DSL) sont par exemple des langages pour l'algèbre linéaire ou des langages de requêtes, ils ne peuvent faire qu'une seule chose, mais ils le font bien. On peut créer et utiliser ces langages *directement dans le C++*, en utilisant la puissance et la flexibilité du C++ pour remplacer les parties communes du langage par les parties spécifiques au domaine que nous utilisons. Dans cette série d'articles, nous allons regarder de près les domain-specific languages, dans quels domaines sont-ils utiles et comment peut-on *facilement* les implémenter en C++ avec l'aide de Boost.Proto (Lien 55).

2. Les langages dédiés

Retour en 1958, John Backus travaillait sur un problème fastidieux, comment précisément établir la syntaxe du langage ALGOL. Il en est arrivé à un système qui décrit la syntaxe de n'importe quel langage informatique. Ce système, qui a beaucoup évolué au fil des années, est connu sous le nom d'EBNF (Lien 56) et il se présente sous diverses formes partout où un langage informatique a besoin d'être défini : spécifications officielles, outils à la construction d'un compilateur, etc.
L'omniprésence d'EBNF est une preuve de sa puissance simple et élégante. Avec juste quelques lignes, on peut, par exemple, décrire la syntaxe, les priorités et l'associativité entre les opérateurs arithmétiques infixes :

```
expression = terme ( ( ?+' terme ) | ( ?-'
terme ) )* ;

terme = facteur ( ( ?*' facteur ) | ( ?/' facteur
) )* ;

facteur = UINT | ?(? expression ?)' | ?-' facteur
;
```

L'EBNF est un excellent exemple de domain-specific language ; c'est exactement l'outil qu'il faut pour ceux qui s'occupent de l'analyse grammaticale et l'analyse syntaxique, mais c'est aussi très utile pour tous les autres. Le fait que ça soit hautement spécialisé fait partie son charme. Il vous permet de vous consacrer pleinement à ce que vous voulez mettre en place, sans être distrait par des choses sans importance. Mais il y a un inconvénient, si vous avez besoin de faire quelque chose qui ne relève pas du domaine de compétence d'EBNF, alors EBNF est inutile. C'est la bête noire de tous les DSL, vous avez souvent besoin d'une petite porte pour vous échapper des limites de votre domaine.
En revanche, le C++ est un langage de programmation polyvalent. Tout ce qui peut être fait peut être fait avec le C++. Mais la puissance et la flexibilité ont un coût : il y a beaucoup de choses pour lesquelles le C++ n'est pas particulièrement à l'aise. La construction des compilateurs me vient à l'esprit, le C++ ne supporte pas directement des choses comme EBNF. Néanmoins, le C++ est très bon pour concevoir des bibliothèques. La responsabilité appartient aux développeurs de bibliothèques pour fournir les meilleures utilisations possible du C++. Et en plaçant les connaissances spécifiques à un domaine dans une bibliothèque C++, les utilisateurs ne tombent pas des nues quand ils ont besoin de faire quelque chose en dehors des compétences de la bibliothèque, ils peuvent toujours retourner au bon vieux C++ !

3. Un langage dans une bibliothèque

Mais quel est le meilleur moyen pour fournir une bibliothèque C++ avec une notation appropriée et des concepts avec lesquels les utilisateurs sont familiers ? Les développeurs de bibliothèques C++ « moderne » sont arrivés avec une solution originale : utiliser les opérateurs fournis par le C++ pour se rapprocher des notations spécifiques à un domaine. Puisqu'ils ressemblent à des petits langages construits de parties d'un plus gros langage, ces bibliothèques sont appelées domain-specific embedded languages que nous abrégerons par DSEL. Si c'est bien fait, les résultats peuvent être impressionnants.
Prenons, par exemple, la bibliothèque Boost.Spirit (Lien 57), une bibliothèque de décomposition analytique qui donne à l'utilisateur un langage semblable à EBNF pour définir la grammaire. Voici l'équivalent avec Spirit de la grammaire EBNF vu plus haut :

```
////////////////////////////////////////////////
//  Une grammaire Spirit qui reconnait les
expressions de calcul infixes
////////////////////////////////////////////////

// boost::spirit::qi fournit le générateur
d'analyse
boost::spirit::qi::rule<Iterator> expression,
term, factor;
```

```
// Une expression est un terme suivi d'un ou
plusieurs zéros
// les autres termes sont séparés par des ?+'
ou ?-'
/* L'équivalent EBNF est :
expression = terme     ( ( '+'    terme ) | ( '-'
terme ) )* ; */
expression = term >> *( ( '+' >> term ) | ( '-'
>> term ) )  ;

// Un terme est un facteur suivi d'un ou
plusieurs zéros.
// Les autres facteurs sont séparés par ?*'
ou ?/'
/* L'équivalent EBNF est :
terme = facteur     ( ( '*'    facteur) | ( '/'
facteur ) )* ; */
term = factor >> *( ( '*' >> factor ) | ( '/' >>
factor ) )  ;

// Un facteur est un entier non signé ou
// une expression entre parenthèses ou
// un facteur négatif
using boost::spirit::qi::uint_;
/* L'équivaleut EBNF est
facteur = UINT  | '('    expression    ')' | '-'
facteur ; */
factor = uint_ | '(' >> expression >> ')' | '-'
>> factor ;
```

Si vous louchez un peu des yeux, vous pourrez voir à quel point la grammaire Spirit correspond à la grammaire EBNF même si cela n'est pas parfait.

Cet exemple illustre parfaitement à la fois la force et la faiblesse des DSEL pour le domaine et plus particulièrement ceux basés sur le C++. Ils sont très bons pour formuler 90 % de votre syntaxe idéale, mais puisque votre DSEL est limité aux opérateurs que Brian Kernighan et Dennis Ritchie nous ont donnés il y a bien longtemps, il reste 10 % qui requièrent un petit peu d'imagination. Certains se contentent des 90 %. D'autres trouvent cela frustrant et prennent un chemin différent. Et c'est OK. Cette série d'articles n'est pas pour ces personnes. Sans rancune.

Une « règle » (rule) dans Spirit est l'équivalent C++ d'une règle de grammaire en EBNF ; les opérateurs >> et | et * ont tous été surchargés et sont utilisés par Spirit pour respectivement définir les séquences de grammaire, l'alternance et la répétition. Comment ça fonctionne ? Un moyen clair pour Spirit pourrait être d'utiliser l'opérateur de décalage de bits à droite >> pour « décaler à droite » et retourner un objet « règle ». À la place Spirit utilise l'opérateur de décalage de bits à droite pour retourner quelques objets temporaires qui tiennent simplement sur les opérandes gauche et droit. Il en est de même pour tous les autres opérateurs. Le résultat est une expression comme :

```
factor >> *( ( '*' >> factor ) | ( '/' >>
factor ) )
```

Ce code « capture » l'expression entière – types, structures, valeurs, etc. – dans un arbre d'expression. Quand cet arbre est assigné à un objet « règle », Spirit parcourt l'arbre et l'interprète comme une règle de grammaire. Quand tout est parcouru, on peut utiliser la variable expression ci-dessus comme un analyseur afin de valider des expressions arithmétiques infixes. Pas mal !
C'est l'approche générale par les DSEL : utiliser la surcharge d'opérateur pour construire des arbres qui capturent l'expression. Après avoir capturé une expression, on peut maintenant la travailler plus en profondeur. À ce moment, un arbre peut être analysé, optimisé, transformé et évalué de manières spécifiques au domaine utilisé.
Spirit est un ensemble d'outils de construction de compilateur comme yacc (Lien 58). En plus de la grammaire EBNF, il fournit d'autres abstractions comme :

- un arbre syntaxique abstrait : un arbre de représentation standard d'un programme analysé ;
- les actions sémantiques : des morceaux de code exécutable que vous pouvez attacher aux règles de grammaire qui font faire des choses à votre analyseur ;
- et une multitude d'algorithmes de compilation qui opèrent sur les AST.

... tous peuvent être exprimés directement dans votre programme C++ comme des expressions. Ces abstractions basiques – grammaires, AST, actions sémantiques et algorithmes de compilation – deviendront importantes plus tard dans un contexte différent que nous verrons.
C'est l'essence même de celui-ci : en installant des choses qui permettent aux concepts du domaine d'être exprimés directement et brièvement avec des expressions C++ (cf. Spirit et EBNF plus haut), on peut rendre les utilisateurs de notre bibliothèque plus à l'aise et donc plus productifs.
Qui plus est, on peut souvent le faire tout en réalisant de grands gains de performances. Avec un accès complet à l'arbre d'expression, une bibliothèque dédiée peut analyser l'arbre et déterminer des stratégies optimales d'évaluation. Un bon exemple est Blitz++ (Lien 59), une bibliothèque de calculs matriciels qui a battu les performances du langage FORTRAN, quelque chose qu'on ne pensait pas possible à l'époque. Il réalise ses performances remarquables en analysant un arbre d'expression complet et en appliquant des transformations complexes et spécifiques comme la fusion de boucles (Lien 60). C'est uniquement possible car Blitz++ a accès à l'arbre expression complet pour effectuer ses optimisations.

4. Les outils de construction de compilateurs

Le terme est « domain-specific embedded language », mais vous êtes pardonnés si vous pensez que Spirit est une simple « bibliothèque ». Imaginez ce qui pourrait changer si nous avions tous cru de tout notre cœur que les DSEL étaient réellement des langages. Suivons cette logique : les bibliothèques qui implémentent les DSEL seraient des compilateurs et écrire une telle bibliothèque serait comme écrire un compilateur. Et comme nous avons vu, écrire des compilateurs est une tâche assez commune avec des outils comme yacc ou Spirit pour rendre le travail plus facile. Il s'ensuit qu'il devrait y avoir un outil pour les développeurs de DSEL, qui fournit des abstractions pour les codeurs : grammaires, AST, actions sémantiques et des algorithmes de compilation. Ce dont nous avons besoin est un ensemble d'outils de construction de compilateurs pour les DSEL. Boost.Proto (Lien 61) est cet ensemble d'outils.
Une analogie avec Spirit est pertinente puisque Spirit et Proto font à peu près la même chose mais à des moments différents. Spirit génère des programmes qui décomposent

du texte à l'exécution. Proto génère des programmes qui analysent des arbres d'expression C++ à la compilation (c'est une propriété curieuse des arbres d'expressions fortement typés, ils ont des structures riches dans leurs types et dans leurs valeurs. Tout effort pour les manipuler signifie que deux calculs parallèles doivent avoir lieu : un sur les types et un sur les valeurs. C'est une complication supplémentaire qu'un bon outil de DSEL devrait supporter sans effort). Pour résumer, un banal compilateur devrait être implémenté avec Spirit. Et Spirit devrait être implémenté avec un outil comme Proto.
Je n'ai pas parlé de la façon dont les bibliothèques sont typiquement implémentées aujourd'hui. Ça implique un bon morceau de métaprogrammation par template (Lien 62) qui est une tâche assez difficile. Proto nous laisse écrire la bibliothèque Spirit en fonction de grammaire semblable à EBNF et d'actions sémantiques, ça ne ressemble plus à l'impénétrable et fragile code spaghetti qui est souvent associé à la métaprogrammation avec des patrons. Ce serait un grand pas en avant :

1. les DSEL seraient plus faciles à développer et maintenir ;
2. les DSEL seraient définis plus rigoureusement et robustement, amenant des interfaces plus conviviales et des meilleurs messages d'erreurs ;
3. et le plus important, les transformations de code qui étaient excessivement complexes avant seraient maintenant réalisables.

Le potentiel pour les transformations de codes complexes ouvre des possibilités infinies. Arrivez-vous à voir les possibilités ? Avec de tels outils, on peut extraire des expressions, analyser des parties et tout ré-assembler pour faire des fusions des boucles pour le calcul matriciel à la manière de Blitz++, ou définir notre propre syntaxe pour des transactions, ou interpréter une expression comme un langage de requête, etc.

5. Bonjour, Boost.Proto

Boost.Proto est présent dans Boost (Lien 63) depuis la version 1.37. Puisque c'est un outil de construction de compilateurs dans la tradition de yacc et Spirit, Proto peut construire un AST pour vous. Avec Proto, c'est aussi simple que ça :

```
#include <iostream>
#include <boost/proto/proto.hpp>
namespace proto = boost::proto;

int main()
{
   // Créons un terminal Proto qui entoure une
chaine de caractères
   // Soyons culotté et appelons le « cout »
   proto::terminal< char const * >::type cout = {
"cout" };

   // Créons un arbre d'expression et  passons-le
à display_expr
   // pour l'afficher joliment.
   proto::display_expr(
       cout << "hello" << ' ' << "proto!"
   );
}
```

L'argument de proto::display_expr n'est pas une expression de sortie, bien que nous n'ayons pas fait exprès d'y ressembler ; à la place, l'expression construit un arbre qui représente l'expression. Qu'est-ce que ça signifie pour les chaînes de caractère et les caractères ? Cela n'a pas d'importance. C'est juste un arbre qui sera évalué plus tard, peu importe la manière ou la quantité. L'opérateur ‹‹ dans l'expression est fourni par Proto, ainsi que tous les autres opérateurs. Construire un AST avec Proto est simple. Vous n'avez pas à définir un patron ou implémenter une surcharge d'opérateur.
Le résultat du programme ci-dessus est :

```
shift_left(
   shift_left(
       shift_left(
           terminal(cout)
         , terminal(hello)
       )
     , terminal( )
   )
 , terminal(proto!)
)
```

Vous pouvez clairement voir que Proto construit un arbre pour nous dont la structure correspond aux priorités et à l'associativité des opérateurs C++.

6. À venir

Comme vous pouvez le supposer, Proto arrive avec des outils pour définir des grammaires qui correspondent aux expressions et des actions sémantiques que vous pouvez associer aux règles de grammaire qui insuffleront la vie à vos DSEL. J'ai tellement de choses à dire, mais je vais garder le reste pour plus tard. Dans les futurs articles, je vais décrire Proto plus en détail, construire de simples mais utiles DSEL tout en travaillant lentement vers un objectif : le design d'un autre DSEL – qui sortira bientôt – appelé Boost.Phoenix, une bibliothèque qui brouille la ligne entre le code C++ et les données, promettant d'apporter la puissance des macros de Lisp (Lien 64) aux programmeurs C++. Quand cette série sera terminée, vous serez comme Néo dans la Matrice (Lien 65), voyant votre propre code comme des données que vous pourrez plier à votre volonté.

Retrouvez l'article d'Eric Niebler traduit par Timothée Bernard en ligne : Lien 66

Programmation fonctionnelle en C++

Tous ceux qui nous lisent ont déjà probablement entendu parler de la « programmation fonctionnelle » présentée comme pouvant apporter des avantages au développement logiciel ou même vantée comme étant un remède miracle. Toutefois, une visite sur Wikipédia pour plus d'informations peut être au premier abord rebutant, avec dès le départ des références au lambda-calcul et aux systèmes formels. Il n'est pas clair immédiatement de voir ce que cela apporte pour écrire de meilleurs logiciels.

1. Introduction

Tous ceux qui nous lisent ont déjà probablement entendu parler de la « programmation fonctionnelle » présentée comme pouvant apporter des avantages au développement logiciel ou même vantée comme étant un remède miracle. Toutefois, une visite sur Wikipédia (Lien 67) pour plus d'informations peut être au premier abord rebutante, avec dès le départ des références au lambda-calcul (Lien 68) et aux systèmes formels (Lien 69). Il n'est pas clair immédiatement de voir ce que cela apporte pour écrire de meilleurs logiciels.

Mon résumé pragmatique : une grande partie des erreurs dans le développement logiciel sont dues aux programmeurs qui ne comprennent pas bien tous les états possibles dans lequel leur code peut s'exécuter. Dans un environnement multithread, le manque de compréhension et les problèmes qui en résultent sont grandement amplifiés, presque sur le point de la panique si vous n'êtes pas attentif. Programmer dans un style fonctionnel rend explicite l'état manipulé par votre code, il devient plus aisé de raisonner à son sujet et, dans un système totalement pur, il ne peut plus y avoir de situation de compétition entre threads.

Je crois qu'il y a une réelle valeur à utiliser la programmation fonctionnelle, mais il serait irresponsable d'encourager tout le monde à abandonner son compilateur C++ et à commencer à coder en Lisp (Lien 70), Haskell, ou, pour être franc, tout autre langage marginal. Au grand dam des concepteurs de langage, il y a de nombreux facteurs externes pouvant être plus importants que les bénéfices d'un langage et le développement de jeux vidéo en présente de nombreux. Nous avons des problèmes multiplateformes, des chaînes d'outils propriétaires, des barrières de certification, des technologies sous licence, des exigences de performance strictes et par-dessus le marché des bases de code ancien ainsi que la disponibilité en main-d'œuvre dont tout le monde fait face.

Si vous êtes dans des circonstances où vous pouvez entreprendre un développement significatif dans un langage non traditionnel, je vous y encourage, mais soyez prêt à prendre quelques coups au nom du progrès. Pour tous les autres : *peu importe dans quel langage vous travaillez, la programmation dans un style fonctionnel offre des avantages*. Vous devriez le faire partout où c'est pratique. Et là où ce n'est pas pratique, vous devriez réfléchir longuement à votre décision. Vous pouvez en apprendre davantage sur les lambdas, les monades, la curryfication, la composition de fonctions évaluées paresseusement sur des ensembles infinis et tous les autres aspects des langages explicitement orientés fonctionnels plus tard si vous le choisissez.

Le C++ n'encourage pas la programmation fonctionnelle, mais il ne l'empêche pas non plus, tout en gardant la possibilité de faire du bas niveau, d'appliquer des instructions SIMD sur des données agencées la main et conservées dans des fichiers mappés en mémoire. Ou toutes autres bonnes choses sérieuses et difficiles dont vous avez besoin.

2. Fonctions pures

Une fonction pure ne porte que sur les paramètres qu'on lui passe et elle se borne à retourner une ou plusieurs valeurs calculées sur la base de ces paramètres. Elle n'a pas d'effets de bord logique. C'est une abstraction, bien sûr ; chaque fonction a des effets de bord au niveau du CPU et la plupart au niveau du tas, mais l'abstraction a toujours de la valeur.

Elle n'accède pas ou ne met pas à jour d'état global. Elle ne conserve pas d'état interne. Elle n'effectue aucune entrée ou sortie. Elle ne modifie aucun des paramètres d'entrée. Idéalement, elle ne prend en paramètre aucune donnée superflue – passer un pointeur allMyGlobals irait à l'encontre du but recherché.

Les fonctions pures ont de nombreuses propriétés intéressantes.

Thread safety. Une fonction pure avec des paramètres passés par valeur est complètement thread-safe. Avec des paramètres passés par référence ou par pointeur, même s'ils sont constants, vous devez être conscient du danger qu'un autre thread effectuant des opérations non pures puisse muter ou libérer ces données, mais ça reste quand même l'un des plus puissants outils à votre disposition pour écrire du code multithread correct.

Vous pouvez trivialement les remplacer par des implémentations parallèles (Lien 71) ou exécuter de multiples implémentations afin de comparer les résultats. L'expérimentation et l'évolution sont beaucoup moins risquées avec des fonctions pures.

Réutilisation. Il est beaucoup plus simple de copier une fonction pure vers un nouvel environnement. Il faut encore se charger des définitions de type et des fonctions pures appelées, mais il n'y a pas d'effet boule de neige. Combien de fois avez-vous eu connaissance d'un bout de code provenant d'un autre système, qui fait ce dont vous avez besoin, mais le sortir de toutes ses dépendances environnementales donnait plus de travail que simplement le réécrire ?

Testabilité. Une fonction pure possède une transparence référentielle, ce qui signifie qu'elle produira toujours le même résultat pour un ensemble de paramètres donné, peu importe le moment auquel elle est appelée, ce qui la rend beaucoup plus facile à utiliser qu'une fonction entremêlée avec d'autres systèmes. Je n'ai jamais été très diligent pour écrire du code de test ; une grande partie du code interagit avec suffisamment de systèmes pour qu'il devienne nécessaire de créer des scripts de test sophistiqués pour fonctionner et je n'ai pas souvent réussi à me convaincre (probablement à tort) que ça en valait la peine. Les fonctions pures sont très simples à tester ; les tests semblent sortir tout droit d'un manuel, où l'on construit des

entrées et examine la sortie. Maintenant, à chaque fois que je tombe sur un bout de code récalcitrant, je le sépare en plusieurs fonctions distinctes pures et écris des tests pour elles. C'est effrayant, mais je trouve souvent quelque chose de faux dans ces situations, ce qui veut probablement dire que je laisse passer trop de choses à travers les mailles du filet.

Compréhensibilité et maintenabilité. Le lien entre les entrées et sorties permet aux fonctions pures d'être plus facile à réapprendre lorsque cela est nécessaire et il y a moins de places pour que des conditions non documentées concernant l'état externe puissent se cacher.

Les systèmes formels et le raisonnement automatisé seront de plus en plus importants à l'avenir. L'analyse de code statique (Lien 72) est importante aujourd'hui et transformer votre code dans un style plus fonctionnel aide les outils d'analyse – ou du moins permet aux outils rapides locaux de concurrencer les outils globaux plus lents et plus coûteux. Nous sommes dans une industrie du « faut que ça soit fait » et je ne vois pas les preuves formelles sur la justesse d'un programme entier devenir un objectif pertinent. Mais être capable de prouver que certaines catégories d'erreurs ne sont pas présentes dans certaines parties d'une base de code sera toujours très utile. Nous pourrions utiliser un peu plus de science et de mathématiques dans nos processus.

Quelqu'un assistant à un cours d'introduction à la programmation pourrait se gratter la tête et penser « tous les programmes ne sont pas censés être écrits comme ça ? ». La réalité, c'est que beaucoup plus de programmes sont des grosses boules de boue (Lien 73) qu'autre chose. Les langages de programmation impérative traditionnels vous donnent des sorties de secours et celles-ci sont utilisées tout le temps. Si vous êtes en train d'écrire du code jetable, faites ce qui est le plus commode, ce qui implique souvent des états globaux. Si vous écrivez du code qui peut être encore en cours d'utilisation un an plus tard, trouvez un équilibre entre ce qui est commode et les difficultés dont vous allez inévitablement souffrir plus tard. La plupart des développeurs ne sont pas très bons pour prédire les souffrances futures que leurs changements vont générer.

3. La pureté en pratique

Tout ne peut pas être pur, à moins que le programme n'opère que sur son propre code source. À un moment donné, vous avez besoin d'interagir avec le monde extérieur. Il peut être amusant, comme lorsqu'on résout un puzzle, d'essayer de pousser la pureté jusqu'à son maximum, mais vient toujours un moment où les effets de bords sont nécessaires et il faut alors les gérer efficacement.

Il n'est même pas nécessaire d'imposer du tout ou rien au niveau de la pureté : une fonction peut être pure à différents degrés et l'écart entre une fonction parfaitement pure et une fonction presque pure est bien moindre que celui qui sépare cette dernière d'une fonction codée en "spaghetti". Faire avancer une fonction vers la pureté améliore le code, même si elle n'atteint pas la pureté complète. Une fonction qui incrémente un compteur global ou vérifie un flag de debug global n'est pas pure, mais si c'est le seul écart, elle bénéficiera tout de même de la plupart des avantages.

Dans un contexte plus large, éviter le pire est généralement plus important qu'atteindre la perfection dans des cas limités. Si vous examinez les fonctions ou les systèmes les plus toxiques que vous avez eu à traiter, ceux dont vous savez qu'ils doivent être manipulés avec des pinces et un masque, il y a fort à parier qu'ils reposent sur un réseau complexe d'état et d'hypothèses dont dépend leur comportement. Et cela ne se limite pas à leurs paramètres. Imposer une certaine discipline dans ces domaines, où tout au moins se battre afin d'éviter que davantage de code se transforme en pagaille similaire, va avoir plus d'impact que de nettoyer quelques routines mathématiques de bas niveau.

Le processus de refactorisation vers la pureté implique généralement de séparer les calculs de l'environnement dans lequel ils opèrent, ce qui signifie quasi invariablement de passer plus de paramètres. Cela peut sembler un peu curieux – une plus grande verbosité dans les langages de programmation est largement conspuée et la programmation fonctionnelle est souvent associée à une réduction de taille de code. Les facteurs qui permettent aux programmes écrits en langages fonctionnels d'être parfois plus concis que les implémentations impératives sont à peu près orthogonales à l'utilisation des fonctions pures – garbage collector, puissants types fondamentaux, pattern matching, list comprehensions, composition de fonctions, différents morceaux de sucre syntaxique, etc. Pour la plupart, ces facteurs de réduction de taille n'ont pas grand-chose à voir avec le fonctionnel et peuvent être trouvés également dans certains langages impératifs.

Vous *devez* vous irriter si vous avez à passer une douzaine de paramètres à une fonction, mais vous pouvez peut-être refactoriser le code d'une manière réduisant la complexité des paramètres.

L'absence de tout support par le langage C++ pour maintenir la pureté n'est pas idéale. Si quelqu'un modifie une fonction basique largement utilisée en la rendant non pure de manière pernicieuse, tous ceux qui utilisent la fonction perdent aussi leur pureté. Cela semble désastreux d'un point de vue des systèmes formels, mais encore une fois, ce n'est pas du tout ou rien, où l'on tombe en disgrâce dès le premier péché. Le développement logiciel à grande échelle est malheureusement statistique.

L'introduction d'un mot-clef « pure » au niveau des langages C et C++ semble très intéressante. Il y a des parallèles étroits avec const – un qualificatif optionnel qui permet la vérification à la compilation de l'intention de programmeur, sans ne jamais faire de mal et qui pourrait souvent aider la génération de codes. Le langage de programmation D offre le mot-clé « pure » : D Programming Language – Functions (Lien 74). Notez la distinction faite entre pureté faible et forte – vous devez également avoir des références d'entrée et de pointeurs const pour être fortement pur.

À certains égards, un mot-clé du langage est trop restrictif – une fonction peut encore être pure, même si elle appelle des fonctions impures, aussi longtemps que les effets de bord ne s'échappent pas de la fonction extérieure. Des programmes entiers peuvent être considérés comme des unités fonctionnelles pures s'ils ne traitent qu'avec des paramètres de la ligne de commande et pas avec des états aléatoires du système de fichiers.

4. Programmation orientée objet

Michael Feathers @mfeathers : la programmation orientée

objet rend le code compréhensible en encapsulant les parties mouvantes. La programmation fonctionnelle rend le code compréhensible en minimisant les parties mouvantes.
Les « parties mouvantes » sont des mutations d'états. Dire à un objet de se changer lui-même, c'est la leçon numéro un dans un livre élémentaire de programmation orientée objet et c'est profondément enraciné chez la plupart des programmeurs, mais c'est un comportement antifonctionnel. Clairement, il y a une certaine valeur dans l'idée de base de la POO qui est de regrouper les fonctions avec les structures de données sur lesquelles elles opèrent. Mais si vous voulez profiter des avantages de la programmation fonctionnelle dans certaines parties de votre code, vous devez faire marche arrière sur certains comportements orientés objet dans ces domaines.
Les méthodes de classe qui ne peuvent pas être constantes ne sont pas pures par définition, parce qu'elles modifient tout ou partie de l'ensemble potentiellement important des états contenus dans l'objet. Elles ne sont pas thread-safe et leurs manières de pousser par petit coup les objets dans des états inattendus sont en effet une source importante de bugs.
Les méthodes objet constantes peuvent encore être techniquement pures si vous ne comptez pas le pointeur implicite const this, mais beaucoup d'objets sont suffisamment larges pour qu'ils constituent une sorte d'état global qui leur est propre, obscurcissant quelques-uns des avantages de la clarté des fonctions pures. Les constructeurs peuvent être des fonctions pures et plus généralement devraient s'efforcer de l'être – ils prennent des arguments et retournent un objet.
Au niveau tactique de la programmation, vous pouvez souvent travailler avec des objets d'une manière plus fonctionnelle, mais cela peut exiger d'en modifier un peu les interfaces. Chez id, nous avons traversé plus d'une décennie avec une classe idVec3 contenant une méthode automodifiante void Normalize(), mais aucune méthode correspondante idVec3 Normalized() const. Beaucoup de méthodes de string avaient été définies de la même manière, en opérant sur elles-mêmes, plutôt qu'en retournant une nouvelle copie avec l'opération effectuée – ToLowerCase(), StripFileExtension(), etc.

5. Implications sur les performances

Dans presque tous les cas, la modification directe de blocs de mémoire est l'approche optimale pour atteindre la vitesse de la lumière et faire autrement consomme un peu de performance. La plupart du temps, cela ne présente qu'un intérêt théorique, car nous échangeons tout le temps des performances contre de la productivité.
La programmation avec des fonctions pures implique plus de copie de données et dans certains cas, cela rend cette stratégie d'implémentation clairement incorrecte pour des raisons de performance. À titre d'exemple extrême, vous pouvez écrire une fonction pure DrawTriangle() qui prend un framebuffer comme paramètre et retourne un framebuffer complètement nouveau avec le triangle dessiné dessus comme résultat. Ne le faites pas.
Tout retourner par valeur est le style naturel en programmation fonctionnelle, mais toujours se fier aux compilateurs pour effectuer la return value optimization (Lien 75) peut être dangereux pour les performances, donc retourner une référence sur des structures de données complexes est souvent justifié. Cela a malheureusement pour effet de vous empêcher de déclarer la valeur retournée comme constante pour imposer l'affectation unique (Lien 76).
Il y aura de fortes envies dans de nombreux cas de simplement mettre à jour une valeur dans une structure complexe, plutôt que de faire une copie de celle-ci et retourner la version modifiée, mais cela ruine les garanties de thread safety et ne doit pas être fait à la légère. La génération de listes est souvent un cas où c'est justifié. La manière purement fonctionnelle d'ajouter quelque chose à une liste est de retourner une copie entièrement nouvelle de la liste plus le nouvel élément à la fin, laissant inchangée la liste originale. Les langages fonctionnels existants sont implémentés de manière à rendre cette manière de faire pas aussi catastrophique que cela puisse paraître, mais si vous le faites avec des conteneurs C++ typiques vous allez mourir.
De nos jours, un facteur atténuant important vient du fait que performance signifie programmation parallèle, ce qui nécessite généralement plus de copie et de combinaison que dans un environnement mono-thread, même dans un cas de performances optimales. La pénalité est donc plus petite, tandis que la réduction de la complexité et les avantages de justesse sont proportionnellement plus grands. Lorsque vous commencez à envisager de déplacer par exemple tous les personnages d'un monde de jeu en parallèle, il commence à s'installer en vous que l'approche orientée objet de mise à jour des objets présente de profondes difficultés dans des environnements parallèles. Peut-être que si tous les objets utilisent simplement une référence vers une version en lecture seule de l'état du monde et que nous copions par-dessus la version mise à jour à la fin de la frame… Hé, attendez une minute…

6. Mesures à prendre

Faites une inspection de certaines fonctions non triviales dans votre base de code et traquez tous les éléments d'état externe auxquels elles peuvent accéder, ainsi que toutes les modifications potentielles qu'elles peuvent faire. Cela donne une remarquable documentation à placer dans un bloc de commentaires, même si vous n'en faites rien avec. Si la fonction peut déclencher, par exemple, une mise à jour de l'écran par le biais de votre système de rendu, vous pouvez jeter vos mains en l'air et déclarer l'ensemble de tous les effets possibles au-delà de la compréhension humaine.
Lors de la prochaine tâche que vous entreprendrez, essayez depuis le début de réfléchir en terme du calcul réel de ce qui se passe. Collectez vos entrées, passez-les à une fonction pure, puis prenez les résultats et faites quelque chose avec.
Lorsque vous déboguez du code, rendez-vous plus conscients des mutations d'états et des paramètres cachés qui jouent à obscurcir ce qui se passe.
Modifiez certains de vos codes d'objet utilitaire pour retourner de nouvelles copies au lieu de s'automodifier et essayez de flanquer du const en face de pratiquement toutes les variables non-itérateur que vous utilisez.

Retrouvez l'article de John Carmack traduit par Arzar en ligne : Lien 77

Faire un jeu vidéo : par où commencer ?

Vous avez une idée de jeu ou vous débutez en programmation et souhaitez faire un jeu, alors cet article est un excellent point de départ pour vous. Ce document regroupe les informations utiles permettant de prendre une décision afin que le démarrage de votre nouveau projet soit le meilleur possible.

1. Introduction

Les jeux vidéo ont toujours été un acteur important dans le monde de l'informatique. Peu de temps après la conception des ordinateurs, des programmes de jeux ont vu le jour. Maintenant, le marché de ce loisir dépasse celui du grand écran.
Tout en jouant, vous avez sûrement eu l'envie de faire votre propre jeu. Ce tutoriel a pour but de poser la première pierre à votre projet afin que celui-ci devienne une réalité.
Ce tutoriel n'est pas écrit que pour les débutants. C'est un guide pour commencer n'importe quel projet de jeu vidéo et rappellera les notions et outils de base permettant de mener votre projet à bien.

1.1. Qu'est-ce qu'un jeu vidéo ?

Avant de commencer, revenons au tout début. Qu'est-ce qu'un jeu vidéo ? C'est en sachant cela qu'il est plus facile d'imaginer les différents éléments nécessaires à sa création.
Tout d'abord, un jeu vidéo n'est pas que l'œuvre d'un seul programmeur (sauf cas très rare). Un jeu vidéo, c'est le rassemblement d'une idée, d'un programme, de dessins et de sons. Ce tout constitue un jeu vidéo. Ces domaines sont eux aussi fragmentés en sous-ensembles :
Programmeur :

- intelligence artificielle - gestion des ennemis et des éléments « neutres » ;
- moteur - bloc qui permet l'orchestration des éléments du jeu ;
- 2D/3D - module qui affiche les images à l'écran ;
- jeu - les règles du jeu, la gestion de la santé, de l'apparition et l'orchestration des éléments et autres ;
- son - module qui gère les sons, la musique selon les événements du jeu ;
- réseau - module permettant de faire des jeux multijoueurs ;
- interface utilisateur - les menus, l'écran de pause, le HUD ;
- outils - l'éditeur de carte et les autres outils nécessaires pour la création du jeu.

Game designer :

- scénariste - créateur de l'histoire, du cadre scénaristique du jeu ;
- gameplay designer - création des règles du jeu (comment gagner/perdre…) ;
- level designer - création des niveaux.

Graphiste :

- artiste 2D ;
- modeleur de personnages ;
- modeleur de décors.

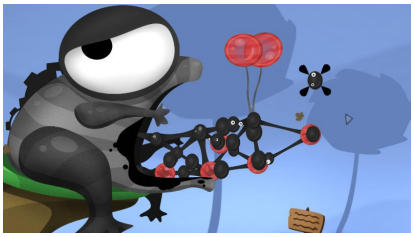
Son :

- effets sonores ;
- musiciens.

Cette liste est loin d'être complète. Par exemple, il peut aussi y avoir des employés à la communication (site web, communauté) ou, pour les projets internationaux, des traducteurs.
Cette liste est vraie pour les très grands projets. Pour de simples projets amateurs, on peut se retrouver à un programmeur et un artiste et cela peut suffire.
Ce qu'il faut comprendre ici, c'est qu'un jeu est avant tout l'histoire d'un(e) ou de plusieurs passionné(e)s. Cela demande énormément de motivation et de temps. Un jeu tel que God of War 3 est réalisé par une équipe complète de professionnels qui ont passé des journées, des semaines, des mois et souvent même des années entières à la conception du jeu.

God of War 3

Mais ce n'est pas pour cela que vous n'allez pas pouvoir faire un jeu. En effet, il est possible de faire des jeux simples qui sont très intéressants. Prenons des exemples comme Minecraft, Binding of Isaac ou World of Goo. Ces jeux ont été réalisés par des passionnés pendant leur temps libre (même s'il est important de rappeler que Notch a décidé d'arrêter son travail pour développer à plein temps Minecraft).



World of Goo

C'est la raison de cet article. Je vais vous guider afin que vous ne tombiez pas dans les pièges classiques d'un projet amateur et que vous prépariez au mieux le développement du jeu avec les outils adéquats pour réaliser vos rêves.

1.2. Faire un jeu vidéo ?

Mais alors, est-ce possible de faire un jeu vidéo tout seul ?
La réponse est oui. Il suffit d'avoir de la motivation, de la patience, du temps ainsi qu'une bonne base au départ (cet article par exemple).
En fait, le principal problème lorsque l'on veut faire un jeu, c'est que l'on se plonge immédiatement sur le projet sans réfléchir. La première étape est de calibrer ses idées à ses possibilités. Ensuite, nous déterminerons comment réaliser ce projet.

1.2.1. Calibrer son projet

En effet, de trop nombreuses fois, j'ai pu lire des messages de personnes cherchant à réaliser un MMORPG du style de World of Warcraft. Certes, c'est un rêve de faire un tel projet, mais, soyons honnête, ce projet est immense et bien trop décourageant pour un petit groupe de personnes.
Pour réussir son projet, il faut le ramener à ses possibilités. Le but est que vous appreniez sur la conception d'un jeu en réalisant des petits jeux (ou brouillons). Ainsi, vous aurez un aperçu des difficultés que vous pouvez avoir mais également un début de réponse à des questions fondamentales.

Que pensez-vous possible pour vous ?

Par exemple, si vous commencez la programmation, il est conseillé de commencer par des projets simples du style Tic-tac-toe, le jeu des allumettes ou la bataille navale (ou encore Mastermind). Oui, ce sont des projets simples, faits déjà mille fois. Mais ne commence-t-on pas chaque langage par un « Hello World » ? Pour ces jeux, je parle d'application en console, toute simple. Par la suite, vous pouvez continuer avec des jeux un peu plus complexes, ou simplement améliorer votre Tic-tac-toe pour lui donner une interface graphique.
Tout projet doit commencer au plus simple. Ensuite, on ajoute des briques, toujours plus complexes, toujours plus grandes. À la fin, on obtient un jeu complet et admirable.
Ne commencez donc pas par un MMORPG, cela serait du suicide et vous découragerait trop tôt (parce qu'en plus, vous ne saurez pas comment le commencer). À la place, faites un jeu en commençant par la base de celui-ci, comme un gestionnaire de combat ou de monstres.

1.2.2. Faire un projet en équipe ?

Selon la taille de votre projet, vous souhaitez ou devez travailler en équipe.
Gérer une équipe demande plus de travail et de préparation sur le projet. En effet, chef de projet est un métier à part. D'autant plus que contrairement à une entreprise, il n'y a pas de motivation d'argent. Chacun aura ses occupations et ses envies, mais aussi son propre emploi du temps rendant la gestion du projet très difficile.
La gestion peut donc être compliquée, mais pas impossible. Premièrement, il faut déterminer l'organisation du travail (qui fait quoi et comment). Ensuite, il faudra continuellement motiver l'équipe et suivre l'avancement des différentes tâches effectuées. Comme c'est votre projet avec votre idée, les autres membres seront moins motivés que vous. Finalement, la coopération et la communication sont des points importants. La distance entre chaque membre ne facilitant pas le travail, il faudra régulièrement faire des réunions à l'aide d'un logiciel tel que Skype. Finalement, pour le développement et pour la gestion du projet, plusieurs outils existent et sont décrits dans la section « les outils ».

1.3. Quelle voie choisir ?

De multiples possibilités s'offrent pour créer votre jeu vidéo. On peut distinguer quatre voies :

- le modding ;
- les game makers ;
- les kits de développement (SDK) ;
- la programmation pure et dure.

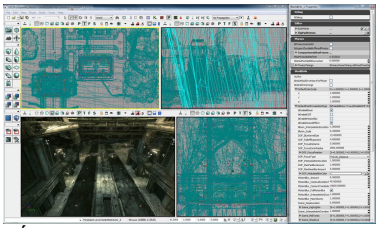
L'ordre choisi pour cette liste prend en compte la nécessité de programmer. Ainsi, le premier élément demande peu ou pas de programmation alors que le dernier consiste uniquement en de la programmation.
Ainsi, il y en a pour tous les goûts. Un artiste voulant tester ses créations pourra le faire très facilement en l'incorporant dans un autre jeu (modding). Un game designer pourra tester ses idées dans un game maker par exemple. Un programmeur pourra, selon son niveau, soit faire son jeu en utilisant un moteur, soit créer directement le sien.

2. Le modding

Le modding est le processus de modification d'un jeu. Certains développeurs proposent des outils pour créer ou éditer le contenu du jeu afin que la communauté puisse étendre son jeu préféré.

2.1. Éditeur

Des jeux comme Skyrim, Starcraft ou Unreal Tournament 3 proposent un éditeur complet afin de créer de nouvelles cartes et même de nouvelles règles de jeu. Chaque développeur de jeu a son propre éditeur. Ainsi, le meilleur moyen d'avoir de l'aide sur ces outils et de lire la documentation et de parcourir les différents sites des communautés de fans.


Éditeur de Unreal Tournament 3

Cette méthode de créer son jeu ne demande aucune ou très peu de connaissances en programmation et permet de s'amuser tout en restant dans l'univers de son jeu préféré.

Un game designer pourra tester ses idées et un artiste pourra intégrer ses nouveaux assets, lorsque cela est possible.

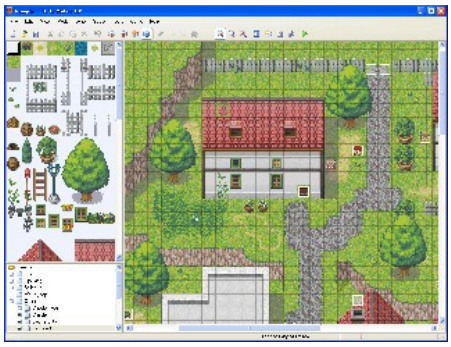
2.2. Modification des fichiers du jeu

Certains jeux ne sont pas accompagnés d'éditeur. Du coup, il faudra modifier directement les fichiers du jeu « à la main ». Vous pouvez néanmoins trouver des logiciels spécialisés sur les sites des fans. Autrement, cela demandera de la recherche, un esprit logique et de la motivation. Toutefois, cela peut être très intéressant, si vous souhaitez savoir comment le jeu fonctionne et si vous aimez tripatouiller.

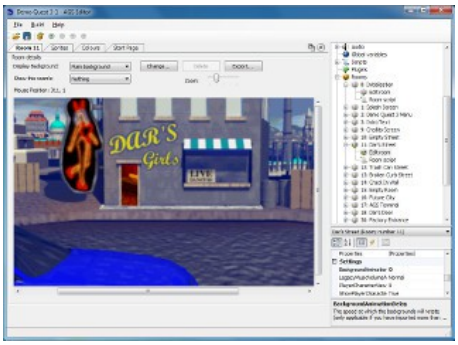
3. Les game makers

Les game makers sont des logiciels aidant à la création d'un jeu d'un type précis mais heureusement, il existe un maker pour chaque type de jeu ou presque. Parmi ceux-ci, nous pouvons citer :

- la gamme RPG Maker pour créer ses propres RPG : Lien 78 ;
- MUGEN, un moteur de jeux de combat : Lien 79 ;
- Adventure Game Studio, un créateur de jeux d'aventure : Lien 80 ;
- 3D Game Studio : Lien 81 ;
- Game Maker : Lien 82.



Chacun d'entre eux propose un ensemble d'outils adaptés et spécialisés pour la création d'un jeu vidéo. Pas ou peu de connaissances en programmation sont nécessaires car ce genre de logiciel réimplémente un langage de scripting très simplifié afin de déclencher des actions dynamiques. De plus, la globalité du jeu peut être créée en cliquant sur des boutons et en configurant des propriétés.



Ce type de logiciel permet de mettre en place ses idées et peut donc être une bonne idée pour tester des mécanismes pour les game designers. Toutefois, si vous souhaitez un jeu qui ne peut se fabriquer dans un moule et qui se veut original par son gameplay, ces logiciels ne vous conviendront certainement pas.

4. Kits de développement (SDK)

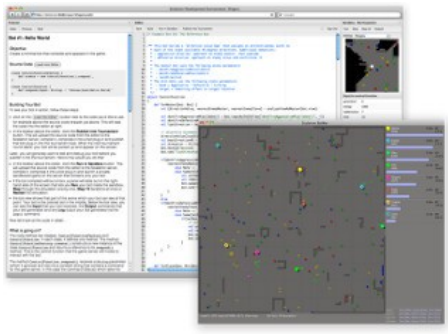
Récemment, plusieurs sociétés de jeux vidéo ont distribué les logiciels qu'ils utilisent durant la conception de leurs jeux vidéo. Ainsi Epic Games (Unreal Tournament, Gears of War) et Crytech (FarCry, Crysis) ont mis à disposition du public l'ensemble des logiciels qu'ils utilisent pour créer leurs jeux.
Mais ce ne sont pas les seuls à avoir proposé une telle suite pour la création de jeux vidéo. En effet, la société Unity Technologies ainsi que quelques autres ont aussi créé leur propre solution, offrant une alternative aux grands noms du milieu à un plus faible coût.
Ainsi, les trois kits les plus célèbres sont :

- Unreal Development Kit (UDK) par Epic Games : Lien 83 ;
- CryENGINE SDK par Crytek : Lien 84 ;
- Unity 3D par Unity Technologies : Lien 85.

Mais il existe aussi d'autres logiciels dans cette lignée, tels que NeoAxis : Lien 86.
Le principal avantage de ce type de solution est la possibilité de créer un jeu rapidement, tout en utilisant les nouvelles techniques et technologies du monde du jeu vidéo.
Toutefois, sachant que l'architecture du jeu est déjà en place, il peut être un peu plus compliqué de faire un jeu dans un style différent de celui pour lequel le kit a été prévu. Cela n'est pas impossible, car les kits laissent une grande liberté, mais pour ce faire il faudra faire de la programmation afin d'implémenter les nouveaux comportements dans le moteur.

Epic Games a créé son propre langage dans l'UDK : l'UnrealScript. Il est avantageux de connaître la programmation avant de partir à la découverte de celui-ci. Unity 3D et NeoAxis se basent sur du C# et le CryENGINE SDK sur du C++.

5. La programmation



Cette section présente les possibilités lorsque l'on souhaite se lancer dans la programmation.

5.1. Le langage

Le premier choix à faire est le langage. En théorie, il est possible de prendre n'importe quel langage pour faire son jeu. En pratique, certains langages seront plus adaptés que d'autres.
Il n'y a pas de langage parfait. Chacun a son point de vue

par rapport aux langages, tiré de sa propre expérience. Donc, le premier conseil, c'est de se faire plaisir.
Vous avez commencé un langage et vous l'aimez bien, alors n'hésitez pas à continuer avec celui-ci. En effet, il est déjà assez compliqué de faire son jeu, alors apporter une nouvelle difficulté en choisissant un langage inconnu n'est sûrement pas une bonne idée.
Certaines plateformes ne supportent pas tous les langages. Ainsi, si vous souhaitez faire un jeu pour des plateformes autres que le PC, renseignez-vous des contraintes pour cette plateforme.
Comme première indication, l'assembleur est fortement déconseillé. Le langage est trop bas niveau pour pouvoir faire des jeux simplement. Seules les vieilles machines (Game Boy, Amiga, Atari) nécessitent encore l'utilisation de ce langage.
Le C et surtout le C++ sont des très bons choix, car couramment utilisés dans le monde du jeu vidéo. De par ce fait, le langage dispose d'une multitude de bibliothèques permettant de réaliser vos jeux. Malheureusement, ce n'est pas le langage le plus simple (même si ce point de vue est relatif) et celui-ci demande une grande rigueur.
Le Java est un langage de choix, car couramment utilisé tout en étant simple. De plus, ce langage est privilégié si vous voulez faire un jeu pour Android.
Le C# est semblable au Java. Toutefois, il a souvent été limité aux plateformes Microsoft. Avec, vous pouvez développer des jeux pour Xbox, Windows Phone et Zune avec XNA, pour navigateur avec Silverlight. Finalement, ce langage est utilisé dans les kits Unity 3D et NeoAxis. Ce langage est fortement utilisé chez les professionnels du jeu vidéo pour la création des outils liés à la conception du jeu (éditeurs, logiciels utilitaires, etc.).
Le BASIC, popularisé car implémenté d'office sur les vieux ordinateurs, est un langage assez simple. Il est toujours possible de faire des jeux avec en utilisant des variantes telles que PureBasic, mais ces solutions ont vieilli.
Le Python est un langage de script simple et puissant. Il est généralement facile à apprendre et est de plus en plus utilisé dans les jeux vidéo pour implémenter la logique du jeu. De plus, il existe de nombreuses bibliothèques aidant à la programmation d'un jeu, comme la très célèbre pygame : Lien 87.

Sachant que les jeux pour le web sont une catégorie bien à part, je vous invite à vous référer à la section Jeux web dans la suite de ce tutoriel.

Ceci est un rapide tour des langages les plus importants, mais il reste plein d'autres solutions.

D'après moi, le Python semble un bon choix pour débuter.

5.2. La bibliothèque

Maintenant que vous avez choisi votre langage, il est nécessaire de déterminer les bibliothèques dont vous avez besoin pour réaliser votre jeu.
Une bibliothèque est une boite contenant une série de fonctions réalisant des tâches simples. Cette boite est nécessaire car il serait très difficile de créer un jeu si nous devions à chaque fois refaire les fonctions de base comme « dessiner un point sur l'écran » ou « ouvrir une fenêtre ».
Comme pour les langages, il existe différents niveaux de bibliothèques. Le bas niveau se retrouve au plus proche de la machine et proposera des fonctions simples du style : « dessiner un point sur l'écran », alors que les bibliothèques de haut niveau proposeront des fonctions permettant de dessiner directement un modèle 3D complet à partir d'un fichier.

Comme il n'est pas possible d'énumérer tous les choix possibles en termes de bibliothèques et de langages, je vais me concentrer sur les bibliothèques les plus courantes en C/C++. Il est possible de découvrir les autres choix sur la page suivante : Lien 88

5.2.1. Bibliothèques bas niveau

Pour la 3D, le choix est assez simple. Il y a OpenGL et DirectX. Le choix se fait selon les plateformes visées. En effet, DirectX n'est supporté que sous Windows et sur Xbox. OpenGL est purement C alors que DirectX est en C++, toutefois, des surcouches (wrappers) permettent d'utiliser ces bibliothèques avec d'autres langages. OpenGL existe dans une variante embarquée : OpenGL ES, utilisée dans les plateformes mobiles (iPhone/iPad/Android).
Alors que OpenGL se limite uniquement à la 3D, DirectX permet aussi de lire les fichiers des modèles 3D, de jouer des sons, de gérer les périphériques de jeu.
Pour la 2D, vous pouvez utiliser la SDL (qui est très portable (compatible même avec la GP2X), mais malheureusement, qui n'a pas évolué depuis longtemps), SFML (moins portable mais à jour et utilisant correctement l'accélération matérielle), Allegro. Plus précisément, ces bibliothèques ne se limitent pas à afficher des images en 2D mais gèrent aussi le son, les périphériques et tout ce qui est nécessaire à la création d'un jeu.

Il est aussi possible de faire de la 2D avec OpenGL et DirectX.

Pour la musique et le son, les solutions sont OpenAL ou directement utiliser les fonctionnalités données par le système d'exploitation. OpenAL permet une spatialisation du son et permettra de jouer les sons dans un environnement 3D.
Pour la physique, la bibliothèque à choisir dépendra de votre projet. En effet, certaines bibliothèques se spécialisent dans la physique dans un monde 2D, alors que d'autres agiront sur un monde 3D. Parmi celles-ci, nous pouvons citer Box2D, Bullet, Newton.
Pour le réseau, les bibliothèques sont liées au système d'exploitation pour lequel vous développez. En effet, chaque système a sa propre implémentation des sockets qui sont la forme la plus basique pour créer une connexion entre deux machines.

Un choix souvent fait par les débutants (choix qui se trouve être un des meilleurs pour faire un jeu rapidement) est de créer son jeu avec SDL et ses modules SDL_mixer, SDL_image et SDL_TTF. Vous pouvez retrouver des explications détaillées dans nos tutoriels : Lien 89.

Bien sûr, ce paragraphe ne peut pas lister toutes les bibliothèques disponibles. Je vous invite donc à visiter notre page Bibliothèques (Lien 90) ou à effectuer des

recherches sur Internet.

5.2.2. Moteurs

Les moteurs sont aussi des bibliothèques mais ceux-ci encapsulent les bibliothèques bas niveau afin de donner un moyen simplifié de faire son jeu. Ainsi, dans les moteurs, des algorithmes avancés seront mis en place vous évitant de devoir les réimplémenter.
Tout comme les bibliothèques bas niveau, les moteurs peuvent se spécialiser :

- **pour la 3D** : Ogre, Irrlicht et OpenSceneGraph ;
- **pour la physique** : ODE, Havok (souvent utilisé dans les jeux vidéo), PhysX et Tokamak ;
- **pour le réseau** : RakNet, HawkNL ou encore GNE ;
- **pour le son** : FModEx ou BASS. FModEx est une bibliothèque couramment utilisée dans le monde des jeux vidéo (Blizzard l'utilise dans Diablo 3 et Starcraft 2).

Cette liste n'est bien sûr pas exhaustive, tant dans les moteurs cités que dans les catégories abordées. On pourra trouver des moteurs un peu plus spécifiques, par exemple des moteurs d'animation 3D, des moteurs de gestion des périphériques, etc.

5.3. Jeux web

Les jeux web sont devenus de plus en plus populaires ces dernières années. En effet, ce succès peut être dû à leur facilité d'accès : il n'est pas nécessaire d'avoir une machine surpuissante, ni même de suivre une procédure d'installation complexe et lente. Il suffit bien souvent de démarrer son navigateur internet pour pouvoir y jouer.

5.3.1. Côté serveur

Le serveur est la machine à laquelle les utilisateurs se connecteront pour jouer. Dans le cas d'un jeu web, le serveur est une machine proposant un site web et est accessible à l'aide d'une URL.
Que vous fassiez un jeu en HTML, en Flash ou avec Unity, si vous voulez le rendre disponible il faudra l'héberger. Cet hébergement est payant et il peut vous en coûter plus ou moins cher selon vos besoins en espace de stockage et en bande passante.
Pour ceux-ci, le site pourra se résumer à sa plus simple forme, une page HTML contenant le code pour importer le module du jeu.
Il existe cependant un autre style de jeu, appelé jeu par navigateur qui repose essentiellement sur l'utilisation du serveur. Dans cette catégorie nous retrouvons les classiques OGame ou Travian.

Ogame

La conception d'un tel jeu est similaire à celle d'un site web. Ainsi, il sera programmé dans un langage comme le PHP, les pages affichées du côté client en HTML/CSS et du JavaScript pour rendre le tout dynamique aux yeux de l'utilisateur.
Maintenant, il existe des frameworks permettant d'éviter le PHP et d'utiliser d'autres langages. C'est le cas pour Django, permettant de faire un site en Python, ou Ruby on Rails, utilisant le Ruby.
Comme vous allez avoir besoin de sauvegarder des données sur les joueurs, les monstres ou autres, il vous sera nécessaire d'avoir une base de données : Lien 91.

5.3.2. Flash

Pendant de nombreuses années, le Flash a été dominant dans les jeux web. En effet, la technologie de Adobe a été la première à proposer de facilement créer des animations et cela de manière indépendante des navigateurs. Le langage de programmation derrière le Flash s'appelle l'ActionScript (actuellement en version 3). La technologie Flash récemment passée en version 11 permet de faire des applications web en 3D.
Pour réaliser une application Flash, plusieurs éditeurs sont disponibles : Lien 92. Adobe développe son propre éditeur : Adobe Flash CS mais celui-ci est payant. Un autre éditeur intéressant est Flash Develop, car celui-ci est gratuit et vous permet pleinement de réaliser un jeu Flash.
Comme le Flash s'est vite popularisé, de multiples bibliothèques de jeux existent et facilitent la création d'un jeu. En effet, à la base le Flash est prévu pour faire de simples animations et n'est pas spécialement conçu pour les jeux. Les bibliothèques comme Flixel permettent d'avoir une gestion simplifiée des composants Flash afin de créer un jeu sans soucis.

D'un autre côté, la technologie semble perdre des adhérents. D'une part, il est possible de faire les mêmes choses avec le HTML 5 et JavaScript, mais en plus Adobe ne supporte plus d'importantes plateformes comme Linux ou Android.

5.3.3. HTML 5/CSS 3

Depuis quelques années, les langages web ont évolué pour offrir une nouvelle dynamique à nos pages. Ainsi, avec un savant mélange de HTML 5/CSS 3 et de JavaScript (et notamment avec Node.js pour le serveur ou encore avec les websockets : Lien 93), il est maintenant aisé de réaliser des jeux complets (Lien 94) et supportés par la majorité des navigateurs et plateformes.
Malgré le jeune âge de ces technologies, différentes bibliothèques existent pour aider à la création d'un jeu : Impact, LimeJS et bien d'autres.
Il existe aussi des logiciels spécialisés comme Construct 2 : Lien 95.

5.3.4. Unity 3D/Unreal Development Kit

Les kits de développement Unity 3D et Unreal Development Kit permettent d'exporter son jeu dans une page web.

5.3.5. Facebook

Le développement de jeux vidéo a atteint une nouvelle

dimension avec l'arrivée des réseaux sociaux tels que Facebook. Ce dernier a mis en place une bibliothèque afin de réaliser des applications pour le réseau et notamment des jeux. Vous pouvez trouver plus d'informations sur le portail de la firme : Lien 96.

5.4. Les outils

Lors de la conception d'un jeu vidéo, il est souvent nécessaire d'utiliser des outils additionnels.
Il vous faudra un outil permettant d'écrire du code. Vous pouvez bien sûr utiliser le bloc-notes, mais comme vous allez sûrement passer beaucoup de temps sur votre code (et que celui-ci n'est pas toujours aussi simple à lire qu'un roman), il est préférable d'utiliser un éditeur spécialisé qui vous apportera des fonctionnalités essentielles telles que la coloration syntaxique. Ensuite, il peut être nécessaire de créer un diagramme UML afin de mettre en place l'architecture de votre jeu. Finalement, et cela devient obligatoire lorsque l'on travaille en groupe, il est nécessaire de mettre en place un logiciel de contrôle de versions.
Ce sont les outils de base pour créer un jeu vidéo. Toutefois, d'autres outils peuvent vous aider, alors n'hésitez pas à jeter un coup d'œil sur cette page : Lien 97.

5.4.1. L'éditeur de code

Un Environnement de Développement Intégré (EDI) est un logiciel regroupant un éditeur de code ainsi que les outils nécessaires pour compiler et produire l'exécutable de ce code. De plus, il contient un débogueur intégré permettant de traquer les bogues de manière simple et efficace.
Pour le C/C++, les EDI les plus connus sont : Microsoft Visual C++, Code::Blocks et QtCreator.
Pour le C#, les EDI sont : Microsoft Visual C# et MonoDevelop.
Pour le Java, on parlera de Eclipse et NetBeans.
Bien sûr, cette liste n'est pas exhaustive, de plus un éditeur peut ne pas se cantonner à un seul langage, tel que Eclipse qui est capable aussi de traiter du C ou encore du PHP. Si vous aimez un autre logiciel et que vous vous sentez bien dans celui-ci, alors ne changez pas.

5.4.2. La modélisation UML

La modélisation est importante afin de ne pas partir tête baissée dans le code et de rencontrer des problèmes d'évolution dans votre logiciel. Bien que la feuille de papier soit le premier outil pour dessiner les diagrammes, il peut être nécessaire d'avoir un logiciel afin de créer son modèle sur ordinateur et de pouvoir ainsi le distribuer aux autres membres de l'équipe.
Les outils disponibles pour créer vos diagrammes sont répertoriés sur cette page : Lien 98.

5.4.3. Logiciels de contrôle de versions

Lorsque l'on travaille à plusieurs sur un même code, il est important de mettre en place un logiciel de contrôle des versions. Celui-ci permet de sauvegarder les différentes modifications de vos fichiers de code et de les gérer. Même si vous travaillez seul, il est conseillé d'en utiliser un afin d'avoir des sauvegardes de votre travail sur un serveur.
Tout d'abord, il existe plusieurs services proposant des serveurs de contrôle de versions (là où votre code sera stocké). Certains de ces services forcent l'utilisation d'une licence où le code doit être ouvert. De plus, le choix d'un de ces services dépendra aussi du protocole que vous voulez utiliser (CVS, SVN, Mercurial, GIT, etc.).
Developpez.com propose ce genre de services, avec un bugtracker et tous les autres outils nécessaires pour faire vivre son projet : Lien 99.
Les autres solutions sont les forges telles que SourceForge, Google Code, Assembla ou GitHub.

4.5.4. Logiciels de suivi des bogues (bugtrackers)

Il peut aussi être intéressant d'utiliser un bugtracker pour suivre l'évolution des bogues ou recevoir les commentaires des utilisateurs de votre jeu. Souvent, les services associés à l'utilisation d'une forge vous donnent l'accès à un bugtracker.
Toutefois, il vous est tout à fait possible de l'installer sur votre propre machine. Les plus connus sont : Mantis et Bugzilla.
Il peut être aussi utilisé pour gérer les différentes tâches du projet et par la même occasion, voir si le projet se déroule correctement.

5.4.5. Logiciels de gestion de projet

Si vous souhaitez une solution plus complète qu'un simple bugtracker, il est possible d'installer un logiciel de gestion de projet. Généralement, ceux-ci se composent d'un bugtracker, d'un wiki, d'un forum, d'une gestion des utilisateurs et de leurs droits sur les projets, d'une intégration avec le gestionnaire de versions et de feuilles de route. Bien entendu, il faudra vous renseigner afin de voir celui qui correspond le mieux à vos attentes.
Parmi ces logiciels, je citerais Redmine et Trac qui ont l'avantage d'être gratuits et de permettre de gérer un projet.

Pour un premier projet, d'autant plus si vous êtes seul et que le projet n'est qu'un simple petit jeu, l'utilisation d'un bugtracker ou d'un logiciel de gestion de projet peut être un peu lourde. Toutefois, cela ne peut qu'être une bonne expérience de découverte. Pour les projets en groupe, le gestionnaire de projet permettra une meilleure communication et un meilleur suivi du projet. Cela permet aussi de garder un historique des discussions et choix effectués pour le projet.

6. La plateforme

Le choix de la plateforme s'est diversifié ces dernières années. En effet, il est devenu facile de développer des applications pour des plateformes autres que le PC.
Dans tous les cas, vous allez effectuer votre développement sur votre PC, car l'écriture du code et la

compilation s'effectuent sur une machine classique.
Il faut savoir que si vous souhaitez cibler une plateforme telle qu'une console ou un smartphone, cela ajoutera un peu de difficulté au développement. Il est donc préférable de commencer avec quelques connaissances en programmation.

6.1. Les plateformes Apple

Tout d'abord, pour développer pour l'iPhone ou l'iPad, il vous faudra un Mac. Des tutoriels sont à votre disposition à cette adresse : Lien 100.
Votre point de départ sera l'installation du SDK : Lien 101.

6.2. Un jeu sur Android

Android est le concurrent direct d'iOS. Vous pouvez trouver plus de tutoriels ici : Lien 102.

6.3. PSP/PS3

Il est aussi possible de développer des jeux pour les plateformes portables de Sony en utilisant des kits non officiels tels que : Lien 103.
Toutefois, il vous faudra un émulateur pour tester vos réalisations (ou encore, en débridant votre console).
Malheureusement, aucun kit officiel n'est disponible gratuitement pour ces consoles. Seule une bêta du kit est disponible : Lien 104.

6.4. Wii/Nintendo DS/Nintendo 3DS

Il est plus simple d'aborder la Nintendo DS que la PSP.
Bien que le kit officiel ne soit pas disponible, il existe plus de ressources sur la question : Lien 105 ou Lien 106.
Il est aussi possible de faire des jeux sur GBA, comme expliqué dans ce tutoriel : Lien 107.
Pour la Wii il existe ces ressources : Lien 108.

6.5. Xbox/Windows Phone

Microsoft est plus ouvert que ses concurrents pour le développement amateur de jeux. La première solution est d'utiliser XNA qui est une bibliothèque de jeux utilisable en C# : Lien 109.
La documentation officielle de Microsoft est très complète sur le sujet, comme vous pouvez en juger : Lien 110.
Il vous est aussi possible d'utiliser Kinect, le capteur de Microsoft, à l'aide du kit officiel : Lien 111.
Avec ce même kit, il est possible de créer des jeux pour Windows Phone.

7. Un choix bien trop souvent oublié ?

Il reste une alternative que l'on écarte/oublie trop souvent. Celle de s'intégrer à un projet Open Source. En effet, l'avantage qu'a un projet Open Source, c'est qu'il est déjà assez avancé pour avoir une certitude de se finir (et d'être présentable). Ainsi, un artiste pourra soumettre ses dessins, un game designer ses idées et un programmeur pourra participer à un projet et rajouter des fonctionnalités intéressantes.

7.1. Licence

Il est très important de lire les documents liés à la licence. En effet, chaque bibliothèque ou logiciel est soumis à des droits d'auteur. La licence permet de déterminer ce qu'il est possible de faire et ce qu'il ne l'est pas.
Les kits de développement (UDK, CryEngine SDK, etc.) possèdent une licence qui permet de réaliser des jeux gratuits, mais dès que vous souhaitez vendre ceux-ci, il faudra payer l'éditeur du kit pour avoir l'autorisation.
Certaines licences pour les bibliothèques peuvent vous obliger à garder votre code ouvert. D'autres n'acceptent pas d'être utilisées dans des applications commerciales.
En réalité, tout est possible en termes de licence. L'utilisation d'un logiciel ou d'une bibliothèque implique que vous acceptiez la licence associée. Donc, lisez bien les licences avant de distribuer votre jeu (le choix des logiciels/bibliothèques est aussi à faire selon la licence).
En conclusion, si vous souhaitez revendre votre projet, ou même, rendre le code disponible librement, il est nécessaire de vérifier que les logiciels et licences des bibliothèques que vous utilisez au sein de votre projet vous laissent la liberté de faire ce que vous souhaitez. Si cela n'est pas le cas, le créateur de la bibliothèque originelle peut toujours demander la fermeture de votre projet.

8. Conclusion

N'hésitez pas à demander de l'aide ou des informations supplémentaires sur la rubrique Jeux (Lien 112) et sur son forum (Lien 113). De plus, comme il est difficile d'être exhaustif sur les technologies permettant de créer des jeux, je ne peux que vous conseiller de jeter un coup d'œil à nos pages des outils (Lien 114) et des bibliothèques (Lien 115).

Retrouvez l'article d'Alexandre Laurent en ligne : Lien 116

Perl CGI et les sessions PHP - Perl/CGI, sessions PHP et cookies

Cet article assez simple a pour but de vous montrer comment échanger des informations entre un programme PHP et un programme Perl/CGI à travers les sessions.

1. Introduction

Beaucoup d'applications/serveurs web ou de sites internet sont développés de nos jours en PHP. Mais ces sites contiennent souvent des pages écrites en Perl/CGI afin d'exécuter des calculs ou d'autres tâches plus facilement gérables via ce formidable langage de programmation. Étant donné que les pages utilisent la plupart du temps des sessions afin de protéger la navigation de chaque utilisateur, le besoin se fait ressentir de pouvoir échanger les sessions entre PHP et Perl.
Cet article va vous expliquer à travers des exemples comment conserver ses variables de session entre ces deux langages.

2. Prérequis

Afin de tester convenablement les exemples de cet article, il est nécessaire que vous disposiez des outils adéquats sur votre PC ou serveur.
Bien sûr, vous devez avoir un serveur web comme Apache par exemple, le langage PHP installé, puis évidemment Perl.
Le but de cet article n'étant pas de vous expliquer comment installer ces outils, je considère que tout est déjà bien installé et configuré pour créer et utiliser des pages web PHP, et qu'Apache est bien configuré pour l'utilisation de programmes Perl/CGI.
Si ce n'est pas le cas, je vous recommande vivement de consulter les documentations sur Developpez.com pour procéder à l'installation et à la configuration des outils.
Une fois tout installé, vous constaterez à la lecture de cet article qu'il est nécessaire d'installer certains modules Perl. Si vous ne savez pas le faire, lisez ce tutoriel vous expliquant comment procéder, quel que soit votre système d'exploitation : Lien 117.

3. Exemple de projet

L'énoncé de notre projet pour l'élaboration de cet article est le suivant :
Dans un premier temps, nos pages seront en PHP puis l'échange se fera avec Perl.
Notre projet contient une page d'accueil qui nécessite une connexion via un identifiant et un mot de passe. Une fois connecté, nous aurons un message d'accueil nous demandant de cliquer sur un lien afin d'avoir la date du jour. Cette date est donnée par un programme Perl. Ensuite, un lien pointe vers une autre page Perl/CGI qui nous renseigne sur la version de Perl.
Le passage d'une page à l'autre se fait grâce à la session créée depuis la page de connexion PHP. Des vérifications sont faites dans chaque page, sous peine soit d'être redirigé vers la page d'accueil, soit d'avoir une page d'informations.
Dans un second temps, la page d'accueil sera réécrite en Perl et le cheminement sera identique. La session sera ainsi créée par Perl et non PHP, puis les échanges se feront dans le sens Perl → PHP → Perl.
Maintenant, à vos claviers !

4. PHP vers Perl

Dans ce chapitre, nous partons du principe que la session a été créée en PHP et l'échange de données se fait avec Perl.
Nos programmes PHP et notre feuille de style CSS sont dans un répertoire « sessions ». L'URL utilisée est de type : **http://localhost/sessions/**.

4.1. Programmes PHP

4.1.1. Page de connexion : index.php

Cette page contient le formulaire de connexion et la première des choses à faire dans notre programme est de créer ou d'ouvrir la session. Je ne vais pas vous expliquer ce qu'est une session. Si vous lisez cet article, c'est que vous le savez !
Pour la création de la session, on utilise la méthode **session_start** en début de programme.
Un fichier dans lequel on trouvera les variables de session sera créé sur le serveur à l'endroit prévu par le fichier de configuration de PHP (**php.ini**) :

```
<?php
  session_start();
?>
```

Étant donné que le lieu de création du fichier de session dépend du serveur web et que nous avons par la suite besoin de le connaître pour un meilleur dialogue avec Perl, nous allons le définir dans notre programme et donner un nom à notre session de la sorte :

```
<?php
// Lieu de la création de la session
session_save_path('C:\TEMP');
// Nom de la session
session_name('SessionPHP');
// Création de la session
session_start();
?>
```

session_save_path lit et/ou modifie le chemin de sauvegarde des sessions : Lien 118.
La méthode **session_name** permet de modifier le nom de la session courante : Lien 119. Ce nom est utilisé comme nom pour les cookies et les URL. Si vous n'en mettez pas

un, celui par défaut sera **PHPSESSID**.
La méthode **session_start()** crée une session ou restaure celle trouvée sur le serveur : Lien 120.
Maintenant, définissons le couple identifiant et mot de passe à utiliser pour valider une connexion :

```
$bon_identifiant = 'test';
$bon_motdepasse  = 'pass';
```

Pour se connecter à la page, il faudra utiliser ces identifiants.
Déclarons maintenant deux variables contenant les identifiants que taperont les utilisateurs sur notre page, avec un clic sur le bouton « Se connecter » :

```
// Bouton connexion pressé
$identifiant = ( isset($_POST['identifiant']) ) ? $_POST['identifiant'] : '';
$motdepasse  = ( isset($_POST['motdepasse'])  ) ? $_POST['motdepasse']  : '';
```

Pour la petite explication, la variable $identifiant contiendra l'identifiant tapé par l'utilisateur et envoyé à la même page via la méthode POST, sinon elle sera égale à une chaîne vide.
Si vous n'avez jamais vu cette écriture, c'est l'équivalent d'un if-else sur une seule ligne via l'opérateur ternaire (« **?:** »).
La première ligne pourrait être écrite de la sorte :

```
if ( isset($_POST['identifiant']) ) {
  $identifiant = $_POST['identifiant'];
}
else {
  $identifiant = '';
}
```

Maintenant, procédons à des tests sur la page.

- Si l'utilisateur a pressé le bouton « Se connecter » et a donné des identifiants erronés :

```
// Bouton connexion pressé et connexion erronée
if ( isset($_POST['connexion']) and ($identifiant != $bon_identifiant or $motdepasse != $bon_motdepasse) ) {
  $message_erreur = "<p class='erreur center'>Erreur de connexion : identifiant ou mot de passe incorrect</p>";
}
```

Dans ce cas, on crée une variable contenant un message d'erreur que nous utiliserons par la suite.

- Si l'utilisateur a pressé le bouton « Se connecter » et a donné des identifiants corrects :

```
// Bouton connexion pressé et connexion correcte
elseif ( isset($_POST['connexion']) and ($identifiant == $bon_identifiant && $motdepasse == $bon_motdepasse) ) {
  $_SESSION["identifiant"] = $identifiant;
  header('Location: http://localhost/sessions/etape1.php');
  exit;
}
```

On redirige l'utilisateur vers le programme PHP suivant via la méthode **header**. L'utilisation d'**exit** est juste une sécurité pour ne pas exécuter la suite du programme.

- Si la session est déjà existante et bonne :

```
// Session déjà existante et bonne
elseif ( isset($_SESSION['identifiant']) && $_SESSION['identifiant'] == $bon_identifiant ) {
 header('Location: http://localhost/sessions/etape1.php');
 exit;
}
```

Idem, redirigeons l'utilisateur.

- Sinon, on affiche le formulaire de connexion :

```
else {
  // Mauvaise session ou bouton connexion non pressé => page normale
}
```

Dans la suite du programme, affichons le formulaire :

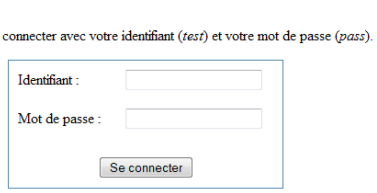
```
<!DOCTYPE HTML>
<html lang="fr">
  <head>
  <meta charset="utf-8">
  <title>Page d'accueil de connexion PHP</title>
  <link href='/sessions/style.css' type="text/css" rel="stylesheet">
  </head>
  <body>
    <h1 class="center">Accueil</h1>
    <p class="center">Merci de vous connecter avec votre identifiant (<i>test</i>) et votre
    mot de passe (<i>pass</i>).</p>

    <form accept="text/html" method="POST" accept-charset="utf-8"
action="http://localhost/sessions/index.php"
      name="formulaire">
    <table class="center">
     <tr>
      <td>Identifiant :</td>
      <td><input name="identifiant" type="text"></td>
     </tr>
     <tr>
      <td>Mot de passe :</td>
      <td><input name="motdepasse" type="password"></td>
     </tr>
     <tr>
      <td colspan="2" style="padding-top:20px;text-align:center;"><input name="connexion" type="submit" value="Se
        connecter"> </td>
     </tr>
    </table>
    </form>
<?php
  if ( ! empty($message_erreur) ) { echo $message_erreur; }
?>
  </body>
</html>
```

C'est du code classique en HTML avec le doctype HTML5 (même si cela n'a aucune importance). Après la balise de fin </form>, nous affichons le message d'erreur d'identification si nécessaire.

Le programme complet **index.php** est le suivant :

```
<?php
// Lieu de la création de la session
session_save_path('C:\TEMP');
// Nom de la session
session_name('SessionPHP');
// Création de la session
session_start();

$bon_identifiant = 'test';
$bon_motdepasse  = 'pass';

// Bouton connexion pressé
$identifiant = ( isset($_POST['identifiant']) ) ?
$_POST['identifiant'] : '';
$motdepasse  = ( isset($_POST['motdepasse'])  ) ?
$_POST['motdepasse']    : '';

// Bouton connexion pressé et connexion erronée
if ( isset($_POST['connexion']) and ($identifiant
!= $bon_identifiant or $motdepasse !=
$bon_motdepasse) ) {
  $message_erreur = "<p class='erreur
center'>Erreur de connexion : identifiant ou mot
de passe incorrect</p>";
}

// Bouton connexion pressé et connexion correcte
elseif ( isset($_POST['connexion']) and
($identifiant == $bon_identifiant && $motdepasse
== $bon_motdepasse) ) {
  $_SESSION["identifiant"] = $identifiant;
  header('Location:
http://localhost/sessions/etape1.php');
  exit;
}
// Session déjà existante et bonne
elseif ( isset($_SESSION['identifiant']) &&
$_SESSION['identifiant'] == $bon_identifiant ) {
 header('Location:
http://localhost/sessions/etape1.php');
 exit;
}
else {
  // Mauvaise session ou bouton connexion non
pressé => page normale
}

?>
<!DOCTYPE HTML>
<html lang="fr">
  <head>
  <meta charset="utf-8">
  <title>Page d'accueil de connexion PHP</title>
  <link href='/sessions/style.css'
type="text/css" rel="stylesheet">
  </head>
  <body>
    <h1 class="center">Accueil</h1>
    <p class="center">Merci de vous connecter
avec votre identifiant (<i>test</i>) et votre
    mot de passe (<i>pass</i>).</p>

    <form accept="text/html" method="POST"
accept-charset="utf-8"
action="http://localhost/sessions/index.php"
      name="formulaire">
    <table class="center">
     <tr>
      <td>Identifiant :</td>
      <td><input name="identifiant"
type="text"></td>
     </tr>
     <tr>
      <td>Mot de passe :</td>
      <td><input name="motdepasse"
type="password"></td>
     </tr>
     <tr>
      <td colspan="2" style="padding-
top:20px;text-align:center;"><input
name="connexion" type="submit" value="Se
        connecter"> </td>
     </tr>
    </table>
    </form>
<?php
  if ( ! empty($message_erreur) ) { echo
$message_erreur; }
?>
  </body>
</html>
```

Voici un exemple de capture d'écran :



4.1.2. Étape 1 : etape1.php

Dans cette page, nous allons afficher l'identifiant, mettre un lien de déconnexion et un lien qui pointe vers le programme Perl pour afficher la date et l'heure.
Comme pour toutes les pages PHP, nous allons ouvrir les sessions et tester si la variable de session stockant l'identifiant est présente ; sinon, on redirige l'utilisateur vers la page d'accueil.
Ensuite, on affiche le contenu de la page :

```
<?php
// Lieu de la création de la session
session_save_path('C:\TEMP');
// Nom de la session
session_name('SessionPHP');
// Création de la session
session_start();

// Vérification de la session
if ( isset($_SESSION["identifiant"]) and
($_SESSION["identifiant"] == "test") )  {
  // OK
}
else {
  header('Location:
http://localhost/sessions/index.php');
  exit;
}

?>
<!DOCTYPE HTML>
<html lang="fr">
  <head>
  <meta charset="utf-8">
```

```
  <title>PHP : &Eacute;tape 1</title>
  <link href='/sessions/style.css'
type="text/css" rel="stylesheet">
  </head>
  <body>
    <h1 class="center">Accueil - &eacute;tape
1</h1>
    <div class="deconnexion"><a
href="/sessions/deconnexion.php">Se
d&eacute;connecter</a></div>
    <p>
    Bonjour <b><?php echo
$_SESSION["identifiant"]; ?></b><br>
    Demandons &agrave; Perl la date du jour : <a
href="http://localhost/cgi-bin/date.pl">ici</a>.

    <hr>
    Variable session : <?php
print_r($_SESSION); ?>

  </body>
</html>
```

On notera que l'on affiche le contenu de la session. Cela permettra d'afficher les variations si besoin :

```
Variable session : <?php print_r($_SESSION); ?>
```

4.1.3. Programme de déconnexion

Ce programme **deconnexion.php** permet d'ouvrir la session et de la détruire. Puis l'utilisateur est redirigé vers la page d'accueil :

```
<?php
  session_save_path('C:\TEMP');
  session_name('SessionPHP');
  session_start();

  // Détruit toutes les variables de session
  $_SESSION = array();

    // Destruction de la session : manière simple
    session_destroy();

    // Redirection en page d'accueil
  header('Refresh: 2;
http://localhost/sessions/index.php');
?>
<!DOCTYPE HTML>
<html lang="fr">
  <head>
  <meta charset="utf-8">
  <title>Deconnexion PHP</title>
  <link href='/sessions/style.css'
type="text/css" rel="stylesheet">
  </head>
  <body>
    <h1 class="center">Déconnexion en
cours...</h1>
  </body>
</html>
```

La destruction de la session entraîne la suppression des variables de session. Ainsi, le fichier sess_xxxxx qui est dans le répertoire C:\Temp sera vidé mais **pas supprimé**. Si vous souhaitez que le fichier soit réellement supprimé, cela dépend de la configuration de PHP (il faut activer le *garbage collector*). Dans le fichier **php.ini**, il faut décommenter et définir la variable session.gc_probability à 1.
Ou bien, dans le programme PHP de déconnexion, mettre : ini_set('session.gc_probability', '1');

4.1.4. Rappel sur les sessions PHP

Dès la session démarrée avec **session_start**, et jusqu'à sa fermeture (typiquement la fin d'exécution du script), PHP pose un verrou exclusif en écriture sur le fichier de session. Durant l'exécution du script, tout processus ayant besoin d'accéder aux variables de session devra attendre que ce verrou soit libéré. En cas de traitement lourd, on peut avoir un blocage suivi d'une erreur de **timeout** des autres processus cherchant à y accéder.

Dans ce cas de figure, il est recommandé de relâcher manuellement ce verrou par un appel à la méthode session_write_close : Lien 121.
N'hésitez pas à vous documenter, il existe de très bons articles sur ce sujet.

4.2. Programme Perl

4.2.1. Programme d'affichage de la date : date.pl

La propagation des sessions se fait en PHP via deux méthodes : par cookies et par URL. La première est celle que nous avons utilisée dans les exemples ci-dessus.
Pour que Perl puisse récupérer la session en cours, il va devoir lire le cookie généré par le programme PHP. Commençons l'écriture de notre programme.
Le but de notre programme est basique : afficher la date. Nous allons le faire via le module **POSIX**. Inutile de chercher à l'installer, il est déjà présent dans le Core de Perl :

Date et heure
```
#!C:/perl/bin/perl.exe
use warnings;
use strict;

use POSIX qw(strftime);

# Date et heure
my $date = strftime "%d/%m/%Y %H:%M:%S",
localtime;
```

Cela affichera une date de ce type : **23/10/2012 22:01:52**
Déclarons maintenant deux variables contenant le nom de la session et le répertoire où elle est stockée :

```
# Nom de la session et répertoire où
sont stockés les fichiers de session.
my $nom_session       = 'SessionPHP';
my $repertoire_session = 'C:\TEMP';
```

Il est maintenant important de récupérer l'identifiant du cookie. Pour cela, nous devons d'abord créer le constructeur CGI qui permet de créer une page web en Perl. Pour cela, il faudra faire appel au module CGI (qui doit être installé sur votre serveur). Je vous recommande également d'installer le module **CGI::Carp** utile en cas de besoin de débogage. Il permet d'afficher les messages d'erreurs sur la page web afin de ne pas toujours aller voir

dans le fichier log.

Pensez à ne plus faire appel au module **CGI::Carp** dans vos programmes en production.

```
use CGI ;
use CGI::Carp 'fatalsToBrowser';
my $cgi = new CGI ;
```

Ensuite, on récupère le cookie :

Cookie

```
# Cookie
my $cookie = $cgi->cookie( -name =>
$nom_session );
```

Il est important de s'assurer que le cookie existe bien et que la session est bien existante, sinon, il ne s'agit pas d'un utilisateur étant passé par le formulaire de connexion vu précédemment. Pour vérifier que la session est existante, il faut tout simplement vérifier qu'un fichier se nommant **sess_IDCOOKIE** est présent dans le répertoire de sessions :

```
# Vérifions l'existence de la session et du
cookie
if ( (not defined $cookie) or (! -e
$fichier_session) ) {
  redirection($cgi);
}

sub redirection {
  my $cgi = shift;
  print $cgi->header(
    -charset => 'utf-8',
    -Refresh => '4;
URL=http://localhost/sessions/index.php',
  );
  print $cgi->start_html(
    -title => "Perl - Etape 2 : interdiction",
    -style => { 'src'=>'/sessions/style.css' },
  );
  print "<p class='erreur center'>La session
<b>$nom_session</b> est inconnue ou
  vous n'avez pas le droit de lire cette
page.</p>";
  print "<p class='center'>Redirection dans
quatre secondes.</p>";
  print $cgi->end_html;
  exit;
}
```

Si le cookie n'existe pas ou si le fichier de session n'existe pas, une redirection est effectuée au bout de quatre secondes. Une procédure de redirection a été créée car elle sera utilisée au moins deux fois dans le programme.
Maintenant, créons la session et/ou assurons-nous que l'identifiant est le bon. Nous aurons besoin du **PHP::Session** (Lien 122) qui permet de lire ou créer une session :

```
use PHP::Session;

# Lecture ou création de la session
my $session = PHP::Session->new( $cookie,
    {
      save_path => $repertoire_session,
      auto_save => 1, # Sauvegarde de la session
en cas de modification
      create => 1,
    }
);
# Variable de session identifiant
my $identifiant  = $session->get('identifiant');
if ( not defined $identifiant or $identifiant ne
'test') {
  redirection($cgi);
}
```

À ce stade, tous les tests ont été effectués, nous pouvons afficher la page. En affichant l'id de la session, la date, puis en mettant un lien qui pointe vers une autre page Perl affichant la version de Perl. De plus, on stockera la date dans la session :

```
print $cgi->header( -charset => 'utf-8', );
print $cgi->start_html(
  -title => 'Perl - Etape 2 : date',
  -style => { 'src' => '/sessions/style.css' },
);

my $id = $session->id;
$session->set( date => $date );

print '<h1 class="center">Perl - &Eacute;tape 2
</h1>';
print '<div class="deconnexion"><a
href="/sessions/deconnexion.php">Se
d&eacute;connecter</a></div>';

print <<"RESUME";
Bonjour <b>$identifiant</b>, aujourd'hui, nous
sommes le : $date<br><br>
Cookie : <b>$cookie</b><br>
Id session : <b>$id</b><br><br>
Perl, affiche ma version : <a href="/cgi-
bin/version-perl.pl">ici</a>
RESUME

print $cgi->end_html;
```

Voici le programme complet :

date.pl

```
#!C:/perl/bin/perl.exe
use warnings;
use strict;
use CGI;
use CGI::Carp 'fatalsToBrowser';
use File::Spec;
use PHP::Session;
use POSIX qw(strftime);

# Date et heure
my $date = strftime "%d/%m/%Y %H:%M:%S",
localtime;

# Nom de la session et répertoire où sont stockés
les fichiers de session.
my $nom_session        = 'SessionPHP';
my $repertoire_session = 'C:\TEMP';

my $cgi = new CGI;

# Cookie
my $cookie = $cgi->cookie( -name =>
$nom_session );
```

```
my $fichier_session = File::Spec->catfile( $repertoire_session, 'sess_' . $cookie );

# Vérifions l'existence de la session et du cookie
if ( ( not defined $cookie ) or ( !-e $fichier_session ) ) {
  redirection($cgi);
}

# Lecture ou création de la session
my $session = PHP::Session->new(
  $cookie,
  { save_path => $repertoire_session,
    auto_save => 1,            # Sauvegarde de la session en cas de modification
    create    => 1,
  }
);

# Variable de session identifiant
my $identifiant = $session->get('identifiant');
if ( not defined $identifiant or $identifiant ne 'test' ) {
  redirection($cgi);
}

print $cgi->header( -charset => 'utf-8', );
print $cgi->start_html(
  -title => 'Perl - Etape 2 : date',
  -style => { 'src' => '/sessions/style.css' },
);

my $id = $session->id;
$session->set( date => $date );

print '<h1 class="center">Perl - &Eacute;tape 2 </h1>';
print '<div class="deconnexion"><a href="/sessions/deconnexion.php">Se d&eacute;connecter</a></div>';

print <<"RESUME";
Bonjour <b>$identifiant</b>, aujourd'hui, nous sommes le : $date<br><br>
Cookie : <b>$cookie</b><br>
Id session : <b>$id</b><br><br>
Perl, affiche ma version : <a href="/cgi-bin/version-perl.pl">ici</a>
RESUME

print $cgi->end_html;

sub redirection {
  my $cgi = shift;
  print $cgi->header(
    -charset => 'utf-8',
    -Refresh => '4; URL=http://localhost/sessions/index.php',
  );
  print $cgi->start_html(
    -title => "Perl - Etape 2 : interdiction",
    -style => { 'src' => '/sessions/style.css' },
  );
  print "<p class='erreur center'>La session <b>$nom_session</b> est inconnue ou
  vous n'avez pas le droit de lire cette page.</p>";
  print "<p class='center'>Redirection dans quatre secondes.</p>";
  print $cgi->end_html;
  exit;
}
```

4.2.2. Programme version-perl.pl

Le programme est semblable à **date.pl**. Seul le contenu affiché de la page change car on affiche la version de Perl. Voici le programme :

version-perl.pl

```
#!C:/perl/bin/perl.exe
use warnings;
use strict;
use CGI;
use CGI::Carp 'fatalsToBrowser';
use File::Spec;
use PHP::Session;
use English '-no_match_vars';

# Nom de la session et répertoire où sont stockés les fichiers de session.
my $nom_session        = 'SessionPHP';
my $repertoire_session = 'C:\TEMP';

my $cgi = new CGI;

# Cookie
my $cookie = $cgi->cookie( -name => $nom_session );
my $fichier_session = File::Spec->catfile( $repertoire_session, 'sess_' . $cookie );

# Vérifions l'existence de la session et du cookie
if ( ( not defined $cookie ) or ( !-e $fichier_session ) ) {
  redirection($cgi);
}

# Lecture ou création de la session
my $session = PHP::Session->new(
  $cookie,
  { save_path => $repertoire_session,
    auto_save => 1,            # Sauvegarde de la session en cas de modification
    create    => 1,
  }
);

# Variable de session identifiant
my $identifiant = $session->get('identifiant');
if ( not defined $identifiant or $identifiant ne 'test' ) {
  redirection($cgi);
}

print $cgi->header( -charset => 'utf-8', );
print $cgi->start_html(
  -title => 'Perl - Etape 3 : Version Perl',
  -style => { 'src' => '/sessions/style.css' },
);

print '<h1 class="center">Perl - &Eacute;tape 3 </h1>';
print '<div class="deconnexion"><a href="/sessions/deconnexion.php">Se d&eacute;connecter</a></div>';
```

```
print <<"RESUME";
Bonjour <b>$identifiant</b>, version de Perl :
<b>$PERL_VERSION</b>
<hr>
<a
href='http://localhost/sessions/etape1.php'>Reven
ir &agrave; l&acute;&eacute;tape 1</a>
RESUME

print $cgi->end_html;

sub redirection {
  my $cgi = shift;
  print $cgi->header(
    -charset => 'utf-8',
    -Refresh => '4;
URL=http://localhost/sessions/index.php',
  );
  print $cgi->start_html(
    -title => "Perl - Etape 2 : interdiction",
    -style => { 'src' => '/sessions/style.css' },
  );
  print "<p class='erreur center'>La session
<b>$nom_session</b> est inconnue ou
  vous n'avez pas le droit de lire cette
page.</p>";
  print "<p class='center'>Redirection dans
quatre secondes.</p>";
  print $cgi->end_html;
  exit;
}
```

On fait appel au module interne de Perl English (Lien 123) (perlvar : Lien 124) qui nous permet d'utiliser la variable spéciale **$PERL_VERSION**.

5. Perl vers PHP

Essayons maintenant de créer *de novo* une session depuis un programme Perl et l'échanger avec un programme PHP. Pour cela, nous allons reproduire la page d'accueil **index.php** vue ci-dessus en Perl.

5.1. Programme accueil.pl

Le programme est assez simple si vous avez bien suivi les premiers chapitres de cet article. Le voici et nous le commenterons par la suite :

accueil.pl

```
#!C:/perl/bin/perl.exe
use warnings;
use strict;
use CGI;
use CGI::Carp 'fatalsToBrowser';
use File::Spec;
use PHP::Session;

# Nom de la session et répertoire où sont stockés
les fichiers de session.
my $nom_session        = 'SessionPHP';
my $repertoire_session = 'C:\TEMP';

my $cgi = new CGI;

# Cookie
my $cookie = $cgi->cookie( -name =>
$nom_session );
my $fichier_session = File::Spec->catfile(
$repertoire_session, 'sess_' . $cookie );

# Lecture ou création de la session
my $session = PHP::Session->new(
  $cookie,
  { save_path => $repertoire_session,
    auto_save => 1,
    create    => 1,
  }
);

my $bon_identifiant = 'test';
my $bon_motdepasse  = 'pass';

# Bouton connexion pressé
my $identifiant    = $cgi->param('identifiant');
my $motdepasse       = $cgi->param('motdepasse');
my $message_erreur = '';

# Bouton connexion pressé et connexion erronée
if (  $cgi->request_method() eq 'POST'
  and $cgi->param('connexion')
  and ( $identifiant ne 'test' or $motdepasse ne
'pass' ) )
{
  $message_erreur
    = "<p class='erreur center'>Erreur de
connexion : identifiant ou mot de passe
incorrect</p>";
}

# Bouton connexion pressé et connexion correcte
elsif ( $cgi->request_method() eq 'POST'
  and $cgi->param('connexion')
  and ( $identifiant eq 'test' or $motdepasse eq
'pass' ) )
{
  $session->set( identifiant => $identifiant );
  print $cgi-
>redirect('http://localhost/sessions/etape1.php')
;
  exit;
}

# Session déjà existante et bonne
elsif ( $session->get('identifiant') and
$session->get('identifiant') eq
$bon_identifiant ) {
  print $cgi-
>redirect('http://localhost/sessions/etape1.php')
;
  exit;
}
else {

  # Mauvaise session ou bouton connexion non
pressé => page normale
}

print $cgi->header( -charset => 'utf-8', -lang =>
'fr' );
print $cgi->start_html(
  -title => "Page d'accueil de connexion Perl",
  -style => { 'src' => '/sessions/style.css' },
);

print <<"RESUME";
    <h1 class="center">Accueil</h1>
    <p class="center">Merci de vous connecter
avec votre identifiant (<i>test</i>) et votre
    mot de passe (<i>pass</i>).</p>
```

```
    <form accept="text/html" method="POST"
accept-charset="utf-8" action="accueil.pl"
name="formulaire">
    <table class="center">
     <tr>
      <td>Identifiant :</td>
      <td><input name="identifiant"
type="text"></td>
     </tr>
     <tr>
      <td>Mot de passe :</td>
      <td><input name="motdepasse"
type="password"></td>
     </tr>
     <tr>
      <td colspan="2" style="padding-
top:20px;text-align:center;">
        <input name="connexion" type="submit"
value="Se connecter">
      </td>
     </tr>
    </table>
    </form>
RESUME

if ( defined $message_erreur ) { print
$message_erreur; }

print $cgi->end_html;
```

Ce script utilise les mêmes principes que dans les programmes Perl ci-dessus. Pour créer les sessions on utilise le module **PHP::Session** : Lien 122.
Dans un premier temps en ligne 16, on essaye de récupérer l'id de la session si celle-ci existe. En ligne 19, on stocke la session ou on la crée si le cookie est à *undef* (l'attribut **create** permet la création si besoin).
De la ligne 27 à 62, nous vérifions si le bouton « Se connecter » a été pressé, si les identifiant et mot de passe saisis sont corrects et surtout si les informations ont été transférées avec la méthode POST. Ou bien, on vérifie que la variable de session est déjà bien présente.
Si tout est OK, on se redirige vers la page PHP (**etape1.php**), sinon, on crée une variable contenant le message d'erreur de connexion.
Par la suite, nous avons le code HTML nécessaire à la création du formulaire de connexion.
Une fois à la page PHP **etape1.php**, le cheminement est celui décrit dans la première partie de cet article. Les sessions sont très bien conservées entre Perl et PHP.

6. Conclusion

Cet article nous a permis de constater qu'il est assez simple de faire communiquer des pages web écrites en PHP et en Perl à travers des sessions. De plus, nous avons l'habitude de concevoir des pages web en PHP et de créer des sessions, mais nous avons vu que la session peut être créée en Perl puis échangée avec PHP. Dans tous les cas, la communication est possible grâce au module **PHP::Session** : Lien 122. Sachez qu'il existe également le module **CGI::Session** (Lien 125) et bien d'autres pour la création de sessions.
Bien évidemment, les sessions ne sont pas faites pour que l'on y stocke des mots de passe, même si c'est possible. Pour cela, vous pouvez tout à fait coupler le tout à une base de données.
L'article a traité le cas d'échanges via des sessions et cookies, mais si les navigateurs bloquent ces popups, il est possible de passer les id des sessions via les URL. Pour en savoir plus, consultez les documentations PHP à ce sujet et celles des modules cités ci-dessus.

7. Liens utiles et programmes à télécharger

Voici la liste des liens utiles de cet article :

- installation des modules Perl CPAN : Lien 117 ;
- cours Apache : Lien 126 ;
- cours PHP : Lien 127 ;
- installer Perl : Lien 128 ;
- les sessions PHP : Lien 129 ;
- modules Perl : CGI (Lien 130), PHP::Session (Lien 122), CGI::Session (Lien 125), English (Lien 131) (perlvar : Lien 124).

Pour tester les programmes de cet article, voici un répertoire contenant les programmes PHP : Lien 132. Le répertoire est à déposer dans votre répertoire **www**. Voici maintenant les programmes Perl/CGI à déposer dans votre répertoire **cgi-bin** : Lien 133. Les répertoires **www** et **cgi-bin** sont les répertoires par défaut, mais tout dépend de la configuration de votre serveur web.

Retrouvez l'article de djibril en ligne : Lien 134.

Chapitre 1 : Déclaration et contrôle d'accès

Le but de cet article est de présenter les notions fondamentales du langage Java de façon concise. Globalement, nous parlerons des énumérations, des classes et des interfaces en prenant le soin de traiter chaque point dans le détails.

1. Introduction

Plusieurs parmi nous (je suis le premier) ont appris le langage Java en rassemblant des bouts de code dans le but de réaliser un logiciel (ce qu'on a coutume d'appeler formation dans le tas). Quoique cela a toujours fini par donner un résultat, on ne saurait cependant renier son inconvénient majeur : à la fin on ne sait jamais comment les choses fonctionnent au fond. Ce qui peut avoir pour conséquence : la mauvaise utilisation des types d'objets, la mauvaise utilisation de la mémoire et parfois occasionne des pertes en temps énormes suite à une mauvaise approche lors du débogage.
L'objectif de ce premier article de la suite le mémo du certifier Java SE 6 (Lien 135) est de mettre à la disposition du développeur l'information succincte concernant les énumérations, des classes, des interfaces, les règles sur les identifiants et le standard JavaBean. À l'issu de cette présentation, cinq exercices corrigés sont mis à la disposition du lecteur pour des besoins d'évaluation personnelle. Je conseille vivement que ce test soit effectué avant et après la lecture de l'article.

2. Énumérations

C'est une structure de données introduite dans la version 5 de Java. Elle permet de construire avec beaucoup plus de simplicité une liste de constantes homogènes.

```
//Exemple de constantes dans une classe
public class Planetes{
  public static final int TERRE = 0;
  public static final int MARS = 1;
  public static final int MERCURE = 2;
  public static final int VENUS = 3;
  public static final int JUPITER = 4;
  public static final int SATURNE = 5;
  public static final int URANUS = 6;
  public static final int NEPTUNE = 7;
}
```

```
//Exemple d'énumération
public enum Planetes{
  TERRE, MARS, MERCURE, VENUS, JUPITER,
  SATURNE, URANUS, NEPTUNE
}
```

2.1. Les règles

2.1.1. La classe Enum

- Une classe de type **enum** est implicitement **final** à moins qu'une des constantes ne contienne un corps de classe. C'est une erreur de compilation d'utiliser le modificateur d'accès "**abstract**" ou "**final**" pour la déclaration d'un **enum**.
- D'après la première règle, on ne peut pas étendre les classes de type **enum.**
- La généricité n'est pas applicable aux classes de type **enum.**

```
public enum Planetes <T> {//erreur de compilation
  TERRE, MARS, MERCURE, VENUS, JUPITER,
SATURNE, URANUS, NEPTUNE;

  T value;
}
```

- Lorsqu'on déclare un type ***enum*** à l'intérieur d'une classe, il est implicitement ***static.***
- D'après la règle précédente, un type ***enum*** ne peut pas être déclaré à l'intérieur d'une classe encastrée (nested class) non ***static.***

```
public class Ecole {
  enum Titre {MONSIEUR, MADAME}//ok

  public class Classe {
     enum Titre {CHEF, SOUS_CHEF}// erreur de
compilation
  }
}
```

- Tous les types ***enum*** sont comparables et sérialisables.
- Une classe de type ***enum*** ne peut pas étendre une autre classe, mais peut implémenter plusieurs interfaces.
- Tous les types ***enum*** étendent la classe java.lang.Enum raison pour laquelle ils ne peuvent pas en étendre une autre.
- *Les classes de type* ***enum*** *contiennent deux méthodes* ***static*** *: values() et valueOf().*

```
static EnumTypeName[] values()
static EnumTypeName valueOf(String name)
```

i. La méthode values() retourne un tableau des constantes d'énumération dans leur ordre de déclaration.

```
enum Planetes{
  TERRE, MARS, MERCURE, VENUS, JUPITER,
  SATURNE, URANUS, NEPTUNE
}
```

```
public class Univers {

  public static void main(String[] args){
     Planetes[] values = Planetes.values();

         for(Planetes value : values){
            System.out.print(value+", ");
         }
  }
}
//output : TERRE, MARS, MERCURE, VENUS, JUPITER,
SATURNE, URANUS, NEPTUNE
```

i. La méthode valueOf(String name) convertit une chaîne de caractère en l'énumération correspondante et génère l'exception IllegalArgumentException si l'argument ne correspond à aucune.

2.1.2. Déclaration

- Une constante d'énumération peut être n'importe quel identifiant Java conforme aux règles.
- Une énumération peut être définie à l'intérieur d'une classe ou dans un fichier comme toutes classes Java.

```
//Enumération dans une classe
public class Univers {
  public enum Planetes{
    TERRE, MARS, MERCURE, VENUS, JUPITER,
    SATURNE, URANUS, NEPTUNE
  }

public static void main(String[] args){
   Planetes[] values = Planetes.values();
        for(Planetes value : values){
            System.out.print(value+", ");
        }
 }
   }
```

```
//Enumération dans le fichier  Planetes.java
 public enum Planetes{
    TERRE, MARS, MERCURE, VENUS,  JUPITER,
SATURNE, URANUS, NEPTUNE
}
```

- Une énumération ne peut pas être définie à l'intérieur d'une méthode.
- ***public*** et ***default*** sont les seuls modificateurs d'accès applicable aux énumérations définies dans un fichier Java (pas à l'intérieur d'une classe).
- L'ordre dans lequel les constantes d'énumérations sont définies compte. Et cet ordre est utilisé dans la comparaison entre deux constantes à l'aide de la méthode compare.
- Il est possible d'ajouter des constructeurs, des variables ou des méthodes dans une classe de type ***enum***.

```
public enum Planetes{
    TERRE(30), MARS(40), MERCURE(50);

        private double rayon;

        Planetes(double rayon){
           this.rayon = rayon;
        }

        public double getRayon(){
           return rayon;
        }
}
```

- Si une classe de type ***enum*** comporte un constructeur, une méthode ou un attribut, les constantes d'énumération doivent être déclarées en premier lieu. Dans le cas contraire une erreur de compilation va se produire.

```
public enum Planetes{
        private double rayon;

   TERRE, MARS, MERCURE;//erreur de compilation
les constantes doivent venir en premier
}
```

- Les énumérations sont des constantes (variables ***final static***) de type ***enum***, de ce fait, elles sont initialisées après le chargement de la classe en mémoire.

```
public enum Planetes{
   TERRE, MARS, MERCURE
}

//son équivalent  en terme de constante Java est
le suivant :

   public enum Planetes{
      public static final Planetes TERRE = new
Planetes();
      public static final Planetes MARS = new
Planetes();
      public static final Planetes MERCURE = new
Planetes();
   }
```

- Il n'est pas obligatoire de mettre un point-virgule à la fin d'une déclaration des énumérations.
- Il est possible de déclarer une méthode abstraite dans une classe de type **enum**, dans ce cas, chaque énumération devrait fournir une implémentation de la méthode sinon une erreur va se produire à la compilation.

```
public enum Planetes{
   TERRE(){ public String getName(){return
"Terre";}},
   MARS(){ public String getName(){return
"Mars";}},
   MERCURE(){ public String getName(){return
"Mercure";}};

   public abstract String getName();
}
```

- Si l'on déclare un attribut, une méthode ou un constructeur dans une classe de type **enum**, la liste des énumérations doit se terminer par un point-virgule.

```
public enum Planetes{
   TERRE, MARS, MERCURE //OK
}

public enum Planetes{
   TERRE, MARS, MERCURE //erreur la liste des
constantes doit s'achever avec un " ; "
   public String rayon;
}

public enum Planetes{
   TERRE, MARS, MERCURE; //OK
   public String rayon;
}
```

2.1.3. Constructeur

- Il n'est pas possible de faire directement appel au constructeur d'une énumération.

```
public enum Planetes{
   TERRE, MARS, MERCURE;

  private double rayon;

  public Planetes(){
     rayon = 0;
  }

  public static void main(String[] args){
     Planetes ptes = new Planetes();//Erreur de
compilation
  }
}
```

- Le constructeur d'une énumération peut avoir comme modificateur d'accès : ***private*** ou ***default***, dans le cas contraire une erreur va se produire à la compilation.
- On peut passer plusieurs paramètres au constructeur d'une classe de type ***enum***

```
public enum Astre {

Mars("planete",12),scrunch("asteroide",1.25f);
    private final float diametre;
    private final String type;
    private Astre(String type,float diametre){
        this.diametre=diametre;
        this.type=type;
    }
}
```

2.2. Quelques méthodes importantes de la classe Enum

```
package java.lang;

public abstract class Enum<E extends Enum<E>>
        implements Comparable<E>, Serializable {

        //Sole constructor.  Programmers cannot
invoke this constructor.
        protected Enum(String name, int ordinal)

        //The name of this enum constant, as
declared in the enum declaration
        private final String name;

        //il retourne la position d'un constant.
Le premier élément est en zéro
        private final int ordinal

        //Returns the name of this enum constant,
exactly as declared in its  enum declaration
        public final String name(){}

        //Returns the ordinal of this enumeration
constant
        public final int ordinal()

        //Returns the enum constant of the
specified enum type with the  specified name
        public static <T extends Enum<T>> T
valueOf(Class<T> enumType,String name)

        public String toString()
}
```

3. Règles sur les identifiants Java

- Un identifiant Java peut être composé à base des caractères Unicodes, des nombres, des symboles de devise ou du caractère underscore (_).

```
Identifiants corrects : number, Number, sum_$,
bingo, $$_100,
                    mål, grüß,$1, ¢d, ¥2, £_
Identifiants incorrects : 48chevy, all@hands,
grand-sum
```

- Un identifiant Java peut débuter par une lettre, un symbole de devise, le caractère underscore, mais jamais par un nombre.
- Après le premier caractère, un identifiant Java peut contenir n'importe quelle combinaison de lettres, de symbole de devis, d'underscores et de nombres.
- Le nombre de caractères qui compose un identifiant n'est pas limité.
- Il n'est pas possible d'utiliser un mot clé Java comme identifiant.

```
public class Planetes{

  public int default = 0;// Erreur de compilation
car default est un mot clé Java
}
```

- Les identifiants Java respectent la casse.

```
public class Planetes{
  //age et Age sont deux identifants Java
différents
  public int age;
  public int Age;
}
```

- Tous les mots clés Java sont en minuscule.
- On crée les constantes en Java en faisant précéder une variable des mots clés ***final*** et ***static***. De plus, la variable doit être de type primitif et sa valeur doit pouvoir être évaluée par le compilateur.

```
//Exemple de constante
    public static final int consommation = 18;
```

```
    public static final String nom = "Rav4";

    //Ces variables ne sont pas des constantes
    public static final String nom = new
String("Rav4");
    public static final char A =
Character.valueOf('A');
```

4. Le standard JavaBean

- Les composants JavaBean sont des classes Java qui respectent certaines règles par rapport à la construction et au nommage des méthodes.
- Comme exemple de règles de construction :

4.1. Règles sur les getters et les setters

- Si la variable n'est pas un booléen, le préfixe d'un getter doit être "***get***".
- Si la variable est un booléen, le préfixe du getter est "**get**" ou "***is***".
- Le préfixe d'un setter doit être "***set***".
- Pour compléter le nom d'un getter ou d'un setter, on transforme la première lettre de la variable en majuscule puis on l'ajoute au préfixe.
- Un setter doit être ***public***, il ne doit retourner aucune valeur (le type retour est ***void***), il prend en paramètre un argument de même type que la variable.
- Un getter doit être ***public***, il ne prend aucun argument et retourne une valeur de même type que l'argument du setter.
- Exemple de composant JavaBean :

```
public class JavaBean {

   private boolean visible;
   private int taille;

   public JavaBean(){}

   public boolean isVisible(){ return visible; }

   public void setVisible(boolean visible)
{ this.visible = visible;}

   public int getTaille(){ return taille}

   public void setTaille(int taille){ this.taille
= taille;}
}
```

4.2. Règles sur les méthodes d'écoutes (les listeners)

Les événements en Java sont des mécanismes permettant aux composants d'envoyer des signaux à d'autres. Ainsi, lorsqu'un composant a besoin d'un signal particulier pour pouvoir accomplir une action, il va demander à se faire enregistrer pour ledit événement. Et à chaque fois que l'événement va se produire, le composant sera informé jusqu'à ce qu'il soit déconnecté. Afin de normaliser les méthodes d'enregistrement et de déconnexion auprès des événements, les conventions de nommage suivantes ont été définies :

- une méthode d'écoute, chargée d'enregistrer un écouteur doit avoir comme préfixe " *add* ", suivi du type de l'écouteur. Exemple : addActionListener() ;
- une méthode d'écoute charger de supprimer un écouteur doit avoir comme préfixe "*remove*", suivi du type de l'écouteur. Exemple removeActionListener() ;
- la méthode d'écoute doit prendre en paramètre un argument du type de l'écouteur à supprimer ou à enregistrer ;
- le nom de la méthode d'écoute doit s'achever par le mot "*Listener*".
- Exemple de classe JavaBean avec méthodes d'écoute :

```
public class JavaBean {

   private boolean visible;
   private int taille;

   private List<ActionListener> listeners = new
ArrayList();

   public JavaBean(){}

   public boolean isVisible(){ return visible; }

   public void setVisible(boolean visible)
{ this.visible = visible;}

   public int getTaille(){ return taille}

   public void setTaille(int taille){ this.taille
= taille;}

   public void addActionListener(ActionListener
istener){listeners.add(listener);}

   public void
removeActionListener(ActionListener listener)
{listeners.remove(listener);}
}
```

5. La déclaration des classes

La notion de classe constitue le fondement de la programmation objet. C'est une structure de données qui permet de définir le type, le comportement et les attributs d'un objet. En Java, on crée une classe à l'aide du mot clé ***class*** suivi du nom de la classe et des accolades.

```
class MaClasseJava {
  // le corps de la classe
}
```

5.1. Règles à propos de la déclaration des classes

- Il peut y avoir uniquement une classe **public** par fichier de code source.
- Les commentaires peuvent apparaître à n'importe quel endroit dans un fichier source.
- Si un fichier source contient une classe ***public***, le nom du fichier doit correspondre au nom de la classe ***public***.
- Si la classe appartient à un ***package***, l'instruction ***package*** doit être la première instruction du fichier avant les instructions ***imports***.
- S'il y a des instructions ***imports***, elles doivent être placées entre l'instruction ***package*** (si le fichier

appartient à un package) et la déclaration de la classe.
- Les instructions ***imports*** et ***package*** sont valables pour toutes les classes contenues dans un fichier source.
- Un fichier source peut contenir plusieurs classes non ***public***.

```
//Le fichier a pour nom Voiture.java
public class Voiture {}

class Totoyota  extends Voiture{}

class Peugeot extends Voiture{}
```

- Un fichier source sans classe ***public*** n'a pas l'obligation d'avoir un nom similaire à l'une des classes qu'il contient.

```
//Le nom du fichier peut être quelconque
exemple : Marques.java
class Voiture {}

class Toyota  extends Voiture{}

class Peugeot extends Voiture{}
```

5.2. Les modificateurs d'accès pour les classes

- Une classe ayant un modificateur d'accès par défaut n'est accessible que par les classes du même package. On l'obtient en ne faisant précéder la déclaration de la classe d'aucun modificateur.

```
public class Voiture {} //modificateur d'accès
public

class Toyota {} //modificateur d'accès par defaut
```

- Une classe avec le modificateur d'accès ***public*** est accessible par toutes les classes de n'importe quels packages.
- Une déclaration de classe précédée du mot clé ***final*** signifie que l'on ne peut pas étendre cette classe.
- Une classe ***abstract*** ne peut pas être instanciée. Sa principale raison d'être se trouve dans l'extension.
- Une méthode ***abstract*** se termine par un point-virgule sans accolades.

```
abstract class Voiture {

   public abstract void rouler();
}

public class Toyota {

   public Toyota(){
      Voiture v = new Voiture();//Erreur à la
compilation
   }
}
```

- Si dans une classe on a une méthode ***abstract***, il faut que la classe soit aussi déclarée ***abstract.***
- Il est possible de mettre des méthodes concrètes dans une classe ***abstract.***
- Une classe ***abstract*** peut avoir un constructeur.
- Toute classe concrète qui étend une classe ***abstract*** doit obligatoirement fournir une implémentation de toutes les méthodes ***abstract***.
- Les combinaisons suivantes sont illégales :

```
static abstract void DoSomething();
strictfp abstract void DoSomething();
native abstract void DoSomething();
synchronized abstract void DoSomething();
private abstract  void DoSomething();
```

6. Règles à propos des interfaces

6.1. Déclaration des interfaces

- Une interface peut étendre une ou plusieurs interfaces.
- Une interface ne peut pas étendre une classe.
- Une interface ne peut pas implémenter (***implements***) une classe ou une autre interface.
- La déclaration d'une interface se fait à l'aide du mot clé ***interface***.
- Une classe qui ***implements*** une interface doit fournir une implémentation concrète de toutes les méthodes contenues dans l'interface. Sinon, la classe doit être déclarée ***abstract***.

6.2. Les variables

- Toutes les variables d'une interface sont implicitement ***public***, ***static***, ***final***. Bref tous les attributs d'une interface sont des constantes.
- Partant de la règle précédente (tous les attributs d'une interface sont des constantes), il en vient que tous les attributs d'une interface doivent être initialisées à la création.
- Une interface ne peut pas contenir une variable non initialisée.

```
public interface MyInterface {
   int value; //Erreur de compilation. car value
est implicitement public static final

   int value1 = 1;//ok
   public int value2 = 1;//ok
   static int value3 = 1;//ok
   final int value4 = 1;//ok
   public static final int value5 = 1;//ok
   static final int value6 = 1;//ok
}
```

6.3. Méthodes

- Une interface peut uniquement avoir des méthodes abstraites (***abstract***).
- Toutes les méthodes d'une interface sont implicitement ***public*** et ***abstract.***
- Partant de la règle précédente (les méthodes d'une interface sont ***public*** et ***abstract***), on déduit que les méthodes d'une interface ne peuvent pas être : ***static***, ***final***, ***strictfp*** ou ***native***.

7. Les modificateurs d'accès

- ***Default*** : On obtient le modificateur d'accès par défaut lorsqu'on ne précise aucun modificateur à la déclaration d'un attribut, d'une méthode ou d'une classe. Un élément précédé du modificateur d'accès par défaut n'est accessible qu'au sein du même package.

```
package com.voiture;
class Toyota {

   int nbreRoues;
   public int nbrePortes;

   int vitesse(){return 17;}
}

/**************************************/

package com.main;
public class Main {

   public static void main(String[] args){
     Toyota t = new Toyota();//Erreur à la
compilation. la classe Toyota n'est pas visible
hors du package
          t.nbreRoues = 4;//erreur de compilation
          t.nbrePortes = 2;//erreur de
compilation
          int i = t.vitesse();//Erreur de
compilation
   }
}
```

- ***Public*** : un attribut ou une méthode ***public*** est accessible par toutes les autres classes qu'elles soient du même package ou pas.
- ***Private*** : un attribut ou une méthode ***private*** n'est accessible qu'au sein de sa classe. Une classe fille ne peut pas hériter des attributs ou des méthodes ***private***.
- ***Protected*** : il est presque identique au modificateur par défaut. Cependant, en plus des classes du même package, les classes dérivées peuvent accéder aux elements ***protected*** d'une classe.
- En général, les modificateurs d'accès ne sont pas applicables aux variables locales à l'exception de ***final*** et ***abstract***.

```
class Toyota {

   private int nbreRoues;

   public void vitesse(){
      final int v = 0;//ok
          abstract class Roue {}//ok

          protected String siege = null;//erreur
de compilation
   }
}
```

Visibilité	Public	Protected	Default	Private
Au sein de la même classe	oui	oui	oui	oui
Au sein du même package	oui	oui	oui	non
Au sein d'une sous-classe dans le même package	oui	oui	oui	non
Au sein d'une sous-classe dans un package différent	oui	par héritage	non	non
Au sein de n'importe quelle classe de n'importe quel package	oui	non	non	non

8. Les éléments d'une classe

8.1. Déclaration des variables

- Il y a deux types de variables en Java :
- Une variable d'instance est une variable définie à l'intérieur d'une classe et en dehors des méthodes ou du constructeur.

```
class Toyota {

   public int nbreRoues;   //variable d'instance

   public Toyota(){
      String nom = "";//variable locale
   }

   public int vitesse(){
      double speed = 0;//variable locale
   }
}
```

- Une variable d'instance peut être : ***public***, ***private***, ***protected***, ***final***, ***transient***. Mais ne peut pas être ***abstract***, ***synchronized***, ***strictfp***, ***static***, ***native.***
- Les variables ***static*** sont des variables de classes pas des variables d'instances. Car une variable ***static*** appartient à toutes les instances d'une même classe.
- Les variables locales sont des variables définies à l'intérieur d'une méthode, d'un constructeur ou d'un bloc d'initialisation.
- Une variable locale doit être initialisée avant d'être utilisée.

```
class Toyota {
   public int vitesse(){
    boolean isOk;
        while(isOk){}//Erreur à la compilation.
isOk doit être initialisée avant toute
utilisation
   }
```

```
}
```

- Une variable ***final*** ne peut pas être initialisée deux fois.

```
class Toyota {

   private final int nbreRoues;
   private final int vitesse = 0;

   public Toyota(){
      nbreRoues = 15;//ok
           vitesse = 12;//erreur de compilation
la variable final a déjà été initialisée
   }
}
```

8.2. Les méthodes à arguments variables

Cette fonctionnalité a été ajoutée à Java 5.0, elle permet aux développeurs de créer des méthodes avec un nombre d'arguments variable. Et se comporte comme un tableau :

```
class Toyota {

   public Toyota(){
      rouler();//ok
        rouler(12);//ok
        rouler(12,0,12);//ok
   }

   public void rouler(int... values){
     for(int i = 0; i < values.length; i++){
              System.out.println("i =
"+values[i]);
          }
    }
 }
```

- Pour définir un argument variable il faut :
- L'argument variable doit être défini en dernière position et on ne peut définir qu'un seul argument variable par méthode.

```
class Toyota {

   public void rouler(int... values){}//ok

   public void rouler(int... values, int speed)
{}//erreur de compilation

   public void rouler(int... values, int...
speed){}//erreur de compilation
}
```

- Une méthode à argument variable peut prendre zéro ou plusieurs arguments.
- Les arguments variables se comportent comme des tableaux. Pour accéder à une valeur ou au nombre d'éléments contenus dans l'argument, on procède de la même façon que dans le cas des tableaux.

8.3. Les éléments static d'une classe

- Les éléments qui peuvent être ***static*** : les méthodes, les variables, les classes encastrées définies en dehors des méthodes, les blocs d'initialisation.
- Les éléments qui ne peuvent pas être ***static*** : les constructeurs, les classes, les interfaces, les variables locales, les classes définies à l'intérieur d'une méthode, les éléments d'une classe encastrée non-static.
- L'encapsulation est un mécanisme Java qui concoure à la maintenabilité, la flexibilité et l'extensibilité des applications. Pour la mettre en œuvre il faut :
- L'accès et la modification des variables à travers des méthodes publiques, permet la modification ultérieure d'une classe sans altérer les classes qui utilisent les services de la classe modifiée.

9. Compatibilité entre les modificateurs d'accès

9.1. Class modifier

	public	abstract	final	strictfp
public		oui	oui	oui
abstract	oui		non	oui
final	oui	non		oui
strictfp	oui	oui	oui	

9.2. Properties (attributes) modifier

Voir le tableau en ligne : Lien 136

9.3. Method modifier

	public	protected	private	static	abstract	final	synchronized	strictfp
native	oui	oui	oui	oui	non	oui	oui	non
static	oui	oui	oui		non	oui	oui	oui
abstract	oui	oui	non	non		non	non	non
final	oui	oui	oui	oui	non		oui	oui
synchronized	oui	oui	oui	oui	non	oui		oui
strictfp	oui	oui	oui	oui	non	oui	oui	

10. Exercices

10.1. Corrigez les erreurs et trouvez le résultat

```
 class Browser {
public static void addToFavorites(int id,
String... values){
System.out.println("id and values");}
public static void addToFavorites(String...
values, int... id){
System.out.println("values and id"); }
public static void addToFavorites(String...
values){
System.out.println("values"); }
public static void main(String... args){
addToFavorites(101);
addToFavorites();
addToFavorites(102,"a");
addToFavorites(103,104,105);
new Browser().addToFavorites(106,"x","y","z");
}
}
```

10.2. Corrigez les erreurs et trouvez le résultat

```
 enum Animals {
DOG("chien"), CAT("chat"), FISH("poisson");
String nom;
public Animals(String s) {nom = s;}
}
public class TestEnum {
static Animals a;
Animals b;
public static void main(String[] args){
System.out.println(a.DOG.nom + " " +a.FISH.nom+ "
" +b.CAT.nom);
System.out.println("ordinals :
"+a.DOG.ordinal());
}
}
```

10.3. Corrigez les erreurs

```
 public class ThisVar {
int planets;
static int suns;
public void gaze() {
int i;
i = this.planets;
i = this.suns;
this = new ThisVar();
this.i = 4;
this.suns = planets; } }
```

10.4. Corrigez les erreurs et trouvez le résultat

```
 package pkgA;
public class Poisson {
int a = 5;
protected int b = 6;
public int c = 7;
private int d = 2;
}
package pkgB;
import pkgA.*;
public class Pecheur {
public static void main(String[] args) {
Poisson p = new Poisson();
System.out.print(" " + p.a);
System.out.print(" " + p.b);
System.out.println(" " + p.c);
System.out.println(" " + p.d);
}
}
```

10.5. Corrigez les erreurs et trouvez le résultat

```
 public class Chat extends Animale {
public static void main(String... args) {
Short _x = 7;
System.out.println(marcher(_x, 6));
}
}
class Animale {
int marcher(int x, int y) { return x + y; }
}
```

11. Correction des exercices

11.1. Exercice 1

- **Erreurs**
  On ne peut définir qu'un seul argument variable par méthode. La méthode public static void addToFavorites(String... values, int... id) produira donc une erreur de compilation. Nous allons la commenter.
  L'argument variable concerne le type String et non les entiers. D'où la ligne addToFavorites(103,104,105); sera commentée.
- **Solution**
  Une fois les corrections faites, nous avons les résultats suivants :

```
 id and values
values
id and values
id and values
```

11.2. Exercice 2

- **Erreurs**
  Si les deux classes sont mises dans le même fichier . java, il faut s'assurer que le nom du fichier est TestEnum, à cause de son modificateur d'accès qui est public.
  Le constructeur d'un type enum ne peut pas être public. D'où, il faut supprimer le modificateur d'accès placer juste avant le constructeur public Animals(String s).
  On ne peut pas directement accéder à une variable de classe dans une méthode static. Il faudra faire précéder la déclaration de la variable Animals b; du mot clé static.
- **Solution**
  Une fois les corrections faites, nous avons les résultats suivants :

```
 chien poisson chat
ordinals : 0
```

11.3. Exercice 3

- **Erreurs**
  La variable this est de type final. De ce fait, on ne peut pas lui affecter une valeur après création. C'est pour cette raison que nous allons commenter la ligne : this = new ThisVar();.
  Le mot clé this permet de référencer l'objet courant. Or, les variables locales ne sont pas accessibles à l'extérieur de la méthode dans laquelle elles ont été déclarées. Ainsi, il est impossible d'indexer une variable locale à l'aide du mot clé this. D'où nous allons commenter la ligne : this.i = 4;.

11.4. Exercice 4

- **Erreurs**
  La variable a n'est précédée d'aucun modificateur d'accès. Raison pour laquelle elle n'est pas accessible en dehors du package

pkgB. Pour résoudre le problème, nous allons soit changer le modificateur d'accès, soit commenter la ligne.
La variable b est précédée du modificateur protected. Ainsi, elle n'est pas accessible à l'extérieur du package pkgB à travers une instance de la classe Poisson. Même solution que la précédente.
La variable d est private, alors elle n'est accessible qu'à l'intérieur de la classe Poisson.

- **Solution**
  Le résultat dépendra des corrections apportées.

11.5. Exercice 5

- **Erreurs**
  Une méthode static ne peut accéder directement à une méthode ou une variable non *static*. C'est le cas avec la méthode marcher(int, int). Pour résoudre le problème, on va faire précéder la méthode marcher(int,int) du mot clé ***static***.
- **Solution**
  Le résultat après correction de l'anomalie est : 13.

Retrouvez l'article d'Armel Fabrice Ndjobo en ligne : Lien 137

Google sort Web Toolkit 2.5 son Framework open source pour le développement JavaScript avec le langage et des outils Java

Google vient de publier la version stable de Google Web Toolkit (GWT) 2.5, son Framework de développement Open Source permettant de créer des applications JavaScript complexes en utilisant le langage et les outils Java.

Le Framework est utilisé en interne par l'éditeur pour le développement de plusieurs de ses produits comme la plateforme publicitaire AdWords et le réseau social Orkut.

GWT 2.5 apporte comme nouveauté phare la fonction Super Dev Mode. Encore au stade expérimental, cette fonction qui remplace Development Mode permet aux développeurs de rapidement recompiler leur code et voir le résultat dans un navigateur. Elle permet également d'utiliser un débogueur pour contrôler une application GWT en cours.

Autre nouveauté : la fonction Elemental, cette bibliothèque expérimentale rapide et légère s'adresse essentiellement aux développeurs qui utilisent les API JavaScript dans leurs applications.

En plus de ces nouveautés, cette version de l'outil apporte de nouvelles optimisations au compilateur GWT, qui permet une réduction de près de 20% de la taille du code généré.

Le compilateur GWT peut désormais utiliser « Closure Compiler » afin de fournir des optimisations JavaScript supplémentaires. Les gros projets qui utilisent Code Splitting et qui ont de nombreux points de partage peuvent profiter de Fragment Merging, qui permet au compilateur de fusionner automatiquement des fragments et réduire la taille de ceux-ci.

On va noter également l'ajout d'une nouvelle bibliothèque pour le support de la norme ARIA (Accessible Rich Internet Applications) du W3C, des améliorations pour UiBinder et de la validation.

Le SDK de GWT 2.5 est téléchargeable gratuitement sous licence Open Source sur le site Google Code, avec un plug-in pour Eclipse.

Télécharger le SDK Google Web Toolkit 2.5 : Lien 138

Commentez la news d'Hinault Romaric en ligne : Lien 139

Support du framework Web Play! sous Netbeans avec le plugin NBPlay

Bonjour tout le monde,

J'ai récemment découvert le framework Play (Lien 140) et je dois dire qu'il m'a tout de suite beaucoup plu. Comme j'utilise NetBeans pour mes développements, j'ai tout naturellement cherché un plugin permettant d'utiliser Play dans NetBeans.
Or il se trouve que Geertjan Wielenga (Lien 141) vient justement de créer le projet NBPlay : Lien 142.

Le projet est tout neuf et ouvert à quiconque souhaite apporter sa contribution. Je me suis moi-même proposé et Geertjan m'a tout simplement ajouté aux committers du projet.

Il n'y a pas encore de build disponible mais vous pouvez à tout moment récupérer les sources via Subversion et les builder avec Maven.

Voici les fonctionnalités implémentées jusque là :

- reconnaissance et ouverture des projets Play ;
- actions clean, build (compile) et run des projets ;
- configuration du répertoire Play dans les options de NetBeans ;
- création de projets Play (Java, Scala, projet vide).

Jonas Fonseca vient également de rejoindre le projet pour travailler sur la partie classpath qui est pour l'instant encore pas mal hardcodée.

Je n'avais encore jamais développé pour la plateforme NetBeans et c'est là une occasion pour moi de la découvrir. J'avoue que ça me plait pas mal, la plateforme est assez facile à prendre en mains et les tutoriels sur netbeans.org sont assez bien faits.

Je vais essayer de bloguer régulièrement sur mes développements relatifs au projet, sur les problèmes que je rencontrerai, etc.

Voilà, n'hésitez pas à faire des commentaires sur le projet. En effet, je ne me suis intéressé à ce framework que récemment et n'ai donc pas vraiment d'expérience quant à son utilisation.
Si vous utilisez Play régulièrement quelles sont, selon vous, les fonctionnalités essentielles à intégrer en priorité ? Toutes les contributions sont les bienvenues.

Voici la liste des billets de Geertjan où il parle de NBPlay (en anglais) :

- Play Framework Plugin for NetBeans IDE (Part 1) : Lien 143
- Play Framework Plugin for NetBeans IDE (Part 2) : Lien 144

Commentez la news de Yann D'Isanto en ligne : Lien 145

Introduction à Xtext

Cet article est le premier d'une série de tutoriels liés à l'utilisation d'Xtext et à la création d'un éditeur pour un DSL personnalisé. Dans ce premier article, nous verrons comment créer une grammaire Xtext et les différentes possibilités qui sont offertes pour rendre l'éditeur généré par Xtext le plus convivial possible, au sein même de la grammaire. Les concepts gravitant autour de la personnalisation de l'IDE non directement liés à la grammaire seront traités dans un autre article.

1. Introduction

Le but de cet article est de présenter le fonctionnement d'Xtext au travers de la mise en place de la grammaire. Tout d'abord nous verrons quels sont les enjeux d'Xtext, puis nous présenterons notre exemple avant de se lancer dans la mise en place de la grammaire. Nous allons aborder dans cet article des notions avancées de gestion de la grammaire Xtext. Pour un tutoriel plus court qui se focalise plus sur les concepts de base, je vous invite à consulter le tutoriel de Georges Kemayo : Introduction à Xtext : créer son propre DSL (Lien 146). D'autre part, seuls les concepts qui ont trait à la grammaire Xtext seront abordés dans cet article. Ceux liés à la personnalisation de l'IDE seront présentés dans un prochain article.

Pour suivre cet article, il est recommandé d'avoir des connaissances en développement de plugins sous Eclipse (Lien 147), ainsi qu'en grammaire des langages (Lien 148). Cet article est basé sur Eclipse 4.2 (Juno).

2. Le framework Xtext

Xtext est un framework pour le développement de langages de programmation et de DSL (Domain-Specific programming Language). Un DSL est un langage propre à un métier ou à une société qui suit un formalisme défini et compact. On peut ensuite faire le pont entre ce langage simple assimilable par tous sans prérequis informatique et un langage de programmation générique, tel que du C ou du Java. Pour plus d'informations reportez-vous à la page Wikipédia : Lien 149. Le framework Xtext s'appuie sur une grammaire générée ANTLR ainsi que sur le framework de modélisation EMF. Ainsi, si vous utilisez déjà une grammaire ANTLR, le bénéfice d'une migration vers Xtext est « limité » à la génération d'un IDE pour votre langage.

Xtext couvre tous les aspects d'un IDE moderne : parseur, compilateur, interpréteur et intégration complète dans l'environnement de développement Eclipse. À ce titre, il permet de couvrir les aspects et les fonctionnalités principales d'un IDE classique.

Xtext est donc adapté aux développeurs désireux de fournir un environnement convivial aux utilisateurs d'un DSL. Il leur permet de mettre en place facilement tous les outils énumérés au paragraphe précédent à partir d'un unique fichier contenant la description de la grammaire du langage.

Parmi les fonctionnalités d'Xtext, on peut mentionner :

- coloration syntaxique : suivant les éléments de la grammaire, Xtext propose une coloration syntaxique entièrement personnalisable pour les éditeurs de votre DSL ;
- autocomplétion : Xtext propose directement l'autocomplétion sur les éléments du langage ;
- validation et « Quick fixes » : Xtext valide le contenu de l'éditeur « à la volée », produisant ainsi un retour direct à l'utilisateur en cas d'erreur de syntaxe. Il permet en plus de proposer, à la manière d'Eclipse, des corrections sur les erreurs détectées ;
- intégration avec d'autres composants Eclipse : en fournissant une API riche pour la manipulation des ressources, Xtext permet de créer facilement des vues graphiques pour la gestion de votre DSL ;
- et plus encore de fonctionnalités pour l'IDE : personnalisation de la vue Outline, recherche des références, hyperliens sont autant d'éléments qui rendent le développement de votre DSL encore plus facile.

Pour aborder tous ces concepts, nous nous baserons sur un exemple simple, décrit dans le paragraphe suivant. Cet exemple (et donc ce tutoriel) met de côté le lien entre la génération d'un DSL et la transformation du DSL en un langage de programmation générique.

3. Présentation de notre DSL

Le but de notre DSL est de créer des fichiers contenant des informations sur des lignes aériennes. Chaque ligne aérienne porte un nom et est définie par :

- un aéroport de départ ;
- un aéroport d'arrivée ;
- un avion ;
- une compagnie ;
- une durée de vol ;
- un paramètre indiquant s'il s'agit d'une ligne régulière.

Un avion est défini par :

- son nom ;
- le nombre de passagers qu'il peut transporter ;
- le nom de la compagnie propriétaire ;
- un paramètre indiquant la motorisation (hélices ou propulseurs).

Un aéroport est défini par :

- son nom ;
- un code OACI ;
- un pays de localisation ;
- un nombre de pistes.

Un exemple de fichier est donné ci-dessous :

```
//Inclusions
%include <avions.air>;
%include <aeroports.air>;

/* ********************* *
   Definition des avions
 * ********************* */
Avion A380:
Passagers: 525;
Motorisation: PROPULSEURS;
End.

Avion ATR42:
Passagers: 48;
Motorisation: HELICES;
End.

/* ************************ *
   Definition des aeroports
 * ************************ */
Aeroport "Marseille-Provence":
OACI: LFML;
Pays: "France";
Pistes: 2;
End.

Aeroport "Roissy-CDG":
OACI: LFPG;
Pays: "France";
Pistes: 4;
End.

Aeroport "JFK-Airport":
OACI: KJFK;
Pays: "USA";
Pistes: 4;
End.

Aeroport "Toulouse-Blagnac":
OACI: LFBO;
Pays: "France";
Pistes: 2;
End.

/* ********************* *
   Definition des lignes
 * ********************* */
Ligne "Toulouse-Marseille":
Avion: ATR42;
Duree: 01h00m;
Compagnie: "Airlinair";
Depart: LFBO;
Arrivee: LFML;
End.

Ligne "NYC-Paris":
Avion: A380;
Duree: 08h15m;
Compagnie: "Air France";
Depart: LFPG;
Arrivee: KJFK;
End.
```

Fichier d'exemple denotre DSL

Comme on peut le voir, le fichier est découpé en quatre blocs.

- Les inclusions : cette section va nous permettre d'inclure des fichiers de notre DSL contenant certaines déclarations. Elle va nous permettre d'alléger le contenu du fichier de déclaration des lignes en déclarant avions et aéroports ailleurs que dans le fichier.
- Les avions : ici sont déclarés les avions. Chaque bloc se termine par « End. », chaque ligne par un ';' et les mots-clés sont séparés des valeurs par ':'. Remarquez que le nom des avions n'est pas entre guillemets. Ceci pour la bonne raison que ces mots ne seront pas considérés comme de simples chaînes de caractères mais bien comme des identifiants de variables.
- Les aéroports : le fonctionnement est le même que pour les avions. De même que pour le nom des avions, c'est ici le code OACI qui va servir de référence.
- Les lignes : dans les déclarations de lignes aériennes, les identifiants des avions et le code OACI des aéroports seront utilisés pour faire les liens entre les variables. Remarquez aussi le format de l'heure, que nous allons spécifier précisément dans notre grammaire.

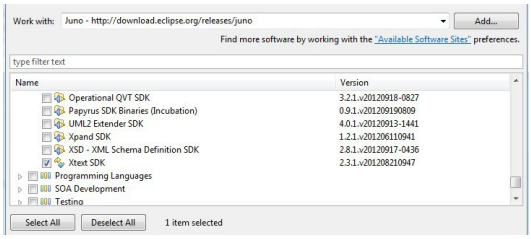
Pour raffiner notre grammaire, nous partirons du postulat que des avions et des aéroports sont déclarés dans le désordre dans un fichier et qu'il se peut que le fichier n'en contienne pas (ils sont alors contenus dans les fichiers inclus). Par contre, les lignes aériennes sont forcément déclarées après les avions et les aéroports. Un fichier peut aussi être vide !

Il est temps de créer notre grammaire Xtext !

4. Création de la grammaire Xtext

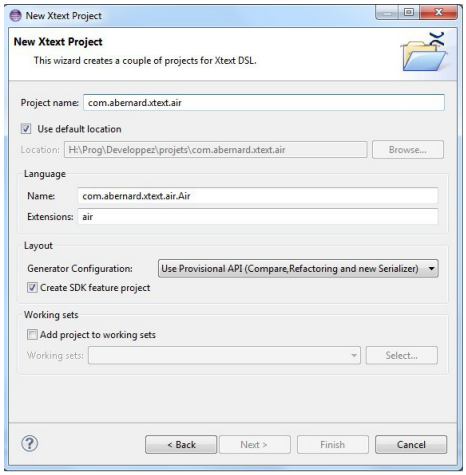
4.1. Installation et démarrage de Xtext

Le plus simple pour commencer à utiliser Xtext est de télécharger la distribution d'Eclipse correspondante sur le site officiel : Lien 150. Cette distribution contient déjà tous les plugins nécessaires au bon fonctionnement d'Xtext (EMF notamment). Néanmoins, si vous souhaitez ajouter Xtext sur une version d'Eclipse existante, parcourez le repository correspondant à votre version d'Eclipse (pour Juno : Lien 151) et sélectionnez « Xtext SDK » :



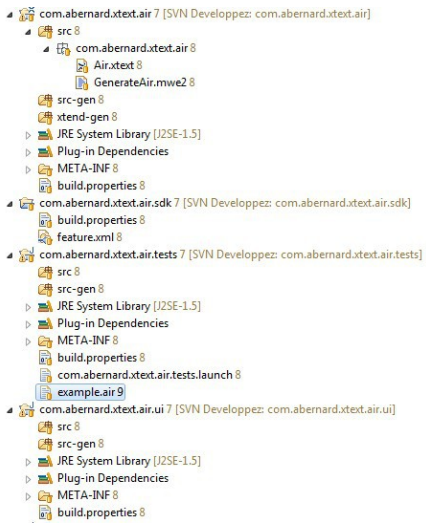
Installation de Xtext

Créez ensuite le projet Xtext via « File > New > Other > Xtext > Xtext project ». Un assistant s'ouvre, vous demandant de renseigner le nom de votre projet mais surtout l'extension de fichier correspondant à votre DSL et le nom de votre DSL. Dans notre cas, nos fichiers porteront l'extension « .air ».



Xtext wizard

L'assistant nous permet aussi de créer directement un projet de type « feature » qui nous permettra de déployer plus facilement nos plugins de DSL. À la fin de l'assistant, quatre nouveaux projets sont créés dans notre workspace Eclipse :



Hiérarchie des projets Xtext

Chacun de ces projets a un but distinct.

- com.abernard.Xtext.air : contient le cœur de notre DSL. Il est pour l'instant vide, hormis deux fichiers : le fichier « Air.Xtext » et le fichier « GenerateAir.mwe2 ». Ces fichiers vont nous permettre respectivement de décrire notre projet et de générer automatiquement toutes les classes de gestion du langage et de l'IDE. Hormis ces deux fichiers, nous ne serons pas amenés à modifier nous-mêmes les fichiers qui seront contenus dans les packages « src », « src-gen » et « xtend-gen ».
- com.abernard.Xtext.sdk : « feature » qui permet de générer facilement les plugins de notre DSL afin de les incorporer dans une distribution Eclipse.
- com.abernard.xtest.air.tests : projet qui peut nous permettre de créer des tests JUnit sur notre grammaire.
- com.abernard.Xtext.air.ui : contiendra l'interface de notre IDE et toutes les classes propres à son fonctionnement. Les packages « src » et « src-gen » seront remplis à la première génération avec des fichiers permettant à Xtext de nous apporter les éléments nécessaires à la personnalisation de notre IDE. Encore une fois, nous ne modifierons pas les fichiers présents dans « src-gen ».

Nos projets étant créés, il nous faut maintenant créer la grammaire Xtext dans le fichier « Air.Xtext ».

4.2. Écriture de la grammaire Xtext

Le contenu du fichier « Air.Xtext » est le suivant pour notre grammaire.

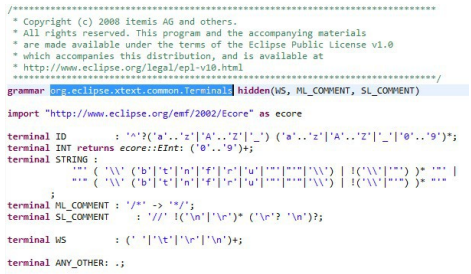
Air.xtext

```
grammar com.abernard.xtext.air.Air with
org.eclipse.xtext.common.Terminals

generate air
"http://www.abernard.com/xtext/air/Air"

// R1 : Racine du fichier
Model:
    (includes+=Includes)*(planes+=Plane |
airports+=Airport)*(airlines+=Airline)*;

// R2 : Declaration des inclusions
Includes:
    '%include''<'importURI=INCLUDE'>;';

// R3 : Declaration des avions
Plane:
    'Avion' name=ID ':'
    ('Passagers'':'passengers=INT ';'
&    'Motorisation'':'motorisation=Engines';')
    'End.';

// R4 : Declaration des aeroports
Airport:
    'Aeroport' title=STRING ':'
    ('OACI'':' name=CodeOACI';'
&    'Pays'':'country=STRING';'
&    'Pistes'':'runways=INT';')
    'End.';

// R5 : Declaration des lignes aeriennes
Airline:
    'Ligne' name=STRING ':'
    ('Avion'':'plane=[Plane]';'
&    'Compagnie'':'company=STRING';'
&    'Depart'':' departure=[CodeOACI|OACI]';'
&    'Arrivee'':'arrival=[CodeOACI|OACI]';'
&    'Duree'':' duration=DURATION';'
&    regular?='REGULIERE;'?)
    'End.';


// R6 : Enumeration des motorisations
enum Engines:
  PROPELLER ='HELICES' | ENGINE = 'PROPULSEURS'
;

// R7 : Duree de vol
terminal DURATION : (('0'..'2''0'..'9')'h')?
('0'..'5''0'..'9')'m';

// R8 : Declaration du code OACI
CodeOACI : name=OACI;

terminal OACI : ('A'..'Z')('A'..'Z')('A'..'Z')
('A'..'Z');

// R9 : Reconnaissance des fichiers inclus
terminal INCLUDE: ID('.air');
```

Nous allons détailler les points particuliers de cette grammaire. Tout d'abord intéressons-nous à la première ligne, qui contient :

```
[\x{0085}] with org.eclipse.Xtext.common.
Terminals
```

Le fichier décrit ici contient un ensemble d'éléments génériques de langage qu'Xtext fournit de manière à simplifier l'écriture de notre grammaire. Ce fichier peut-être ouvert grâce à un Ctrl-Clic.



Xtext Terminals

4.2.1. « Hidden terminals »

La première ligne contient le texte :

```
hidden (WS, ML_COMMENT, SL_COMMENT)
```

Cela énumère les éléments de la grammaire qu'Xtext ne prendra pas en compte dans les analyses syntaxiques et sémantiques. Typiquement, les commentaires sur une ligne (SL_COMMENT) et sur plusieurs lignes (ML_COMMENT). De même les espaces et les retours chariot (WS) n'ont pas d'importance sémantique.

La règle SL_COMMENT a une importance toute particulière : en s'assurant de sa présence dans votre grammaire, vous débloquerez automatiquement la fonctionnalité Eclipse pour commenter/décommenter un bloc, grâce au raccourci Ctrl+Maj+C ou à un clic droit puis « Toggle Comment » dans l'éditeur.

4.2.2. Terminaux

Xtext définit aussi une manière générique de décrire un identifiant (de variable par exemple) grâce à la règle ID. Il propose aussi une définition pour les entiers (INT) et pour les chaînes de caractères (STRING). Tous ces éléments sont des éléments dits « terminaux ». Leur nom est systématiquement en majuscules et ils ne peuvent contenir d'affectation. Ce sont des valeurs brutes caractérisées par une suite de caractères ou un intervalle délimité par des expressions (caractérisé par '->', dans ML_COMMENT). On peut par contre spécifier le type de données qu'auront ces terminaux dans le modèle généré. Regardons par exemple :

```
terminal INT returns ecore::EInt:
```

Cette instruction signifie que les terminaux INT auront une valeur de type EInt, élément définissant les entiers dans le métamodèle ECore. L'utilisation de ECore est permise par l'instruction :

```
import "http://www.eclipse.org/emf/2002/Ecore" as ecore
```

On peut aussi si on le souhaite inclure ecore dans notre propre grammaire afin d'utiliser les types de données proposés. Notons que les cardinalités des différents éléments dans ces terminaux sont données par des expressions de type EBNF : Extended Backus-Naur Form. De fait les cardinalités suivantes peuvent être exprimées :

- exactement un (par défaut) ;
- un ou zéro (opérateur '?') ;
- indéfini (opérateur '*') ;
- au moins un (opérateur '+').

Au sein des terminaux, on peut utiliser différents mécanismes tels que les intervalles de caractères ('0'..'9'), les intervalles de texte (grâce au caractère '->'), les « wildcards » ('f'.'o' va reconnaître fAo, fOo, f_o\x{0085}) et les négations exprimées grâce à '!'. **Il est important de noter que l'ordre des terminaux est essentiel.** En effet lors de la lecture du fichier, Xtext s'arrête dès qu'il a trouvé un terminal correspondant au texte lu. Dès lors, les terminaux de la grammaire peuvent se masquer les uns les autres. Il est donc recommandé si possible d'utiliser des éléments de type DataType.

4.2.3. DataTypes

Ces éléments sont décrits comme des règles standards sauf que, comme les terminaux, aucune assignation ne peut être effectuée. Voici par exemple la déclaration d'un flottant utilisant cette technique :

```
Double returns ecore::EDouble : INT('.'INT)?;
```

Notons que si aucun type de retour n'est indiqué, Xtext retournera par défaut un ecore::EString.

4.2.4. Types énumérés

Il est possible de construire des énumérations. Par exemple, la visibilité d'un package en Java serait décrite sous la forme d'une énumération (trois états : private, public, protected). Dans notre grammaire, le type de motorisation se prête bien à une énumération : regardons la règle R6 des motorisations.

```
// R6 : Enumeration des motorisations
enum Engines:
  PROPELLER ='HELICES' | ENGINE = 'PROPULSEURS'
;
```

Nous définissons ici une énumération à deux éléments, PROPELLER ou ENGINE. L'utilisation dans le modèle de ces éléments sera effective suivant le texte rentré par l'utilisateur, respectivement 'HELICES' ou 'PROPULSEURS'.

4.2.5. Assignations

Lors de la construction des règles de notre grammaire, nous pouvons assigner les règles et les valeurs entrées dans l'éditeur à des variables qui seront présentes dans notre modèle. Ces assignations peuvent être de trois sortes :

- affectation simple : du type maVariable = Regle : ici l'objet recevra simplement la valeur retournée par la règle. Dans notre grammaire, cela est visible par exemple dans la règle R3 : « Avion name=ID ». Ici la variable « name » aura la valeur de l'ID entré dans l'éditeur. Notons à ce sujet que les variables nommées « name » ont une signification particulière dans Xtext : le framework assimilera automatiquement cette variable au nom de l'objet, qui sera affiché dans la vue Outline par exemple ;
- affectation multiple : du type maListe += Regle : ici l'objet maListe sera une liste de Regle. Autant de fois la règle apparaîtra dans le document, autant il y aura d'éléments dans la liste. Par

exemple, dans la règle R1 « includes+=Includes », la variable « includes » contiendra la liste de tous les fichiers déclarés comme inclus dans ce fichier ;
- affectation conditionnelle : du type monBooleen ? = 'TEXTE' : ici le booléen recevra vrai ou faux suivant la présence ou non du texte indiqué à droite du signe '='. Par exemple dans la règle R5 « regular?='REGULIERE';' », la variable « regular » recevra « true » si l'utilisateur entre 'REGULIERE' dans son fichier.

Il est aussi possible d'écrire des règles sans assignation. Par exemple :

```
Regle : RegleA | RegleB | RegleC;
```

Il est parfois nécessaire de forcer l'instanciation d'un objet du modèle. Par exemple, si vous entrez la règle suivante :

```
Regle : (regle=Element)?;
```

Xtext vous affichera alors l'avertissement suivant :

```
The rule 'Regle' may be consumed without object
instantiation. Add an action to ensure object
creation, e.g. '{Regle}'.
```

Il suffit alors comme suggéré d'ajouter « {Regle} » juste après les ':', ce qui permettra d'avoir toujours une instance de Regle créée, même si aucun texte n'a été entré pour cette règle :

```
Regle : {Regle}(regle=Element)?;
```

4.2.6. Références

Xtext est capable de créer directement les hyperliens entre la déclaration des variables de votre DSL et leur utilisation dans les fichiers. Pour cela, un mécanisme de déclaration de références est utilisé, signalé par le positionnement de l'objet à lire entre crochets. Il est important de retenir que :
- seule la référence est entre crochets, pas la déclaration elle-même ;
- l'élément entre crochets fait référence à un objet de type EClass et non à une règle grammaticale ! ;
- par défaut, le terminal de l'élément entre crochets est considéré comme ID. Le terminal indique en fait le texte qui va être concrètement remplacé par le lien. On peut spécifier un autre terminal en ajoutant un pipe ('|') derrière l'élément, suivi du terminal désiré.

Dans notre grammaire, deux types de références existent : les références aux codes OACI des aéroports et aux noms des avions. Observons d'abord le cas le plus simple : les avions. La règle R3 « Plane » va implicitement créer le type Plane qui comporte un identifiant de type ID. Nous pouvons alors dans la déclaration des lignes aériennes créer une référence vers le type Plane grâce à l'instruction « 'Avion':'plane=[Plane]';' ». Comme mentionné précédemment, inutile de spécifier un type particulier car par défaut le type cherché est 'ID'.
Dans le cas des aéroports, le terminal n'est pas ID mais de type OACI. Il faut donc préciser le type de terminal qu'Xtext doit rechercher afin de créer les liens entre données, sous la forme « airport=[CodeOACI|OACI]; » :

```
// R5 : Declaration des lignes aeriennes
Airline:
    'Ligne' name=STRING ':'
    ('Avion'':'plane=[Plane]';'
&   'Compagnie'':'company=STRING';'
&   'Depart'':' departure=[CodeOACI|OACI]';'
&   'Arrivee'':'arrival=[CodeOACI|OACI]';'
&   'Duree'':' duration=DURATION';'
&   regular?='REGULIERE;'?)
    'End.'
;
```

Déclaration des références

4.2.7. Imports de fichiers

Xtext fournit des outils pour gérer les imports de fichiers. À ce titre, le framework est capable de gérer les références que nous venons de décrire au travers de plusieurs fichiers, grâce à un mécanisme d'import de fichiers par leur URI. Pour indiquer à Xtext quels éléments correspondent à des imports, il suffit de les marquer avec l'attribut « importURI », comme dans notre règle R2 « '%include"<'importURI=INCLUDE'>;' ». Parfois ce mécanisme nécessite d'être affiné mais cela sera étudié dans le prochain article.

4.2.8. Règles de grammaire

D'autres règles existent, en plus de celles citées dans les paragraphes précédents. Tout d'abord, on peut spécifier que l'ordre des règles est indéfini, mais que les règles doivent apparaître au moins une fois. Dans l'ensemble de nos déclarations par exemple, la liste des paramètres n'a pas un ordre défini, mais tous doivent être présents. Cela est réalisé en séparant les règles par le symbole '&'. Un autre exemple simple est celui des préfixes de méthodes et d'attributs en Java : on peut écrire 'public static final' ou 'final public static'.

Enfin, on peut aussi forcer la reconnaissance d'une règle pour lever par exemple une indétermination. En effet, dans le cas de conditionnelles imbriquées telles que :

```
if (condition())
    if (autreCondition())
    maMethode();
    else
    autreMethode();
```

Dans un cas tel que celui-ci, impossible de prédire avec précision à quel 'if' appartient le 'else'. Xtext propose de spécifier au parser quelle règle doit être consommée en priorité, comme le montre l'exemple ci-dessous :

```
Condition:
    'if' '(' condition=BooleanExpression ')'
    then=Expression
    (=>'else' else=Expression)?
;
```

Nous avons vu dans cette partie les éléments essentiels à la construction d'une grammaire Xtext assez fournie et complexe. Pour toute référence supplémentaire, je ne peux qu'à vous enjoindre de consulter directement la documentation officielle (Lien 152), plus fournie. Une fois

notre fichier créé, il est temps de passer à la génération de la grammaire.

4.3. Génération de l'infrastructure associée à la grammaire

4.3.1. Le fichier MWE2

Le fichier « GenerateAir.mwe2 » situé dans le même dossier que le fichier « Air.Xtext » définit la liste des fragments qui vont être utilisés par Xtext pour la génération de l'éditeur. Le tableau suivant récapitule certains fragments et leur fonction :

Fragment	Éléments générés
EcoreGeneratorFragment	Code EMF pour les modèles générés
XtextAntlrGeneratorFragment	Grammaire ANTLR, parser, lexer, services associés et modèle Ecore
GrammarAccessFragment	Accès à la grammaire dans le code
ResourceFactoryFragment	Constructeur de ressources EMF
ParseTreeConstructorFragment	Sérialisation modèle vers texte
ImportNamespacesScopingFragment	Gestion du scope des imports
JavaValidatorFragment	Validation de modèle
FormatterFragment	Formateur de code
LabelProviderFragment	Label Provider
OutlineTreeProviderFragment	Construction de la vue Outline
JavaBasedContentAssistFragment	Autocomplétion basée sur Java
XtextAntlrUiGeneratorFragment	Autocomplétion basée sur ANTLR

On peut choisir les fragments que l'on souhaite générer. D'autres fragments sont présents, par exemple pour le support des tests JUnit, le refactoring ou la génération d'un wizard de création de projets pour nos fichiers. Plus particulièrement, les sections qui vont nous intéresser sont les sections de gestion des imports et des liens.

```
// scoping and exporting API
// fragment = scoping.ImportURIScopingFragment {}
// fragment = exporting.SimpleNamesFragment {}

// scoping and exporting API
fragment = scoping.ImportNamespacesScopingFragment {}
fragment = exporting.QualifiedNamesFragment {}
fragment = builder.BuilderIntegrationFragment {}
```

Valeurs par défaut

Par défaut, les liens sont recherchés à travers tous les fichiers existants, ce qui pose problème pour une gestion claire et propre des fichiers inclus. Nous allons donc modifier cette section de la manière suivante, afin que la création des liens se fasse au sein d'un unique fichier et de ses imports :

```
// scoping and exporting API
fragment = scoping.ImportURIScopingFragment {}
// fragment = exporting.SimpleNamesFragment {}

// scoping and exporting API
// fragment = scoping.ImportNamespacesScopingFragment {}
fragment = exporting.QualifiedNamesFragment {}
fragment = builder.BuilderIntegrationFragment {}
```

Nouvelles valeurs

4.3.2. Lancement de la génération

Une fois ces deux fichiers préparés, nous pouvons lancer la génération de notre grammaire. Ceci peut se faire de deux façons :

- soit par clic droit sur le fichier Xtext > « Run As > Generate Xtext Artifacts »
- soit par clic droit sur le fichier MWE2 > « Run As > MWE2 Workflow »

Cette action va lancer la génération de tous les éléments nécessaires à notre éditeur. Cette étape peut être longue, suivant la taille de la grammaire. La première fois qu'on lance la génération, le message suivant apparaît dans la console Eclipse :

```
*ATTENTION* It is recommended to use the ANTLR 3
parser generator (BSD licence -
http://www.antlr.org/license.html). Do you agree
to download it (size 1MB) from
'http://download.itemis.com/antlr-generator-
3.2.0.jar'? (type 'y' or 'n' and hit enter)
```

Il suffit de répondre 'y' et la génération continue. Ce JAR ne peut pas être inclus par défaut avec Xtext pour des problèmes de licence : le code généré par ANTLR et utilisé par Xtext n'est pas contraint en licence. Il n'utilise pas une « librairie/runtime » (par exemple) qui pourrait avoir une licence contaminante. À la fin de la génération, tous les dossiers « src-gen » de vos plugins devraient être fournis avec un certain nombre de classes. Dans le plugin « ui », le dossier « src » contient le package « org.Xtext.ui » qui nous permettra de personnaliser notre IDE.

4.3.3. Résoudre les erreurs : LL-recursivity

Lorsque Xtext rencontre un problème lors de la génération de la grammaire, c'est quasiment systématiquement suite à la vérification de celle-ci par le parser ANTLR. Si la plupart des erreurs sont compréhensibles et donc assez faciles à corriger, il en est une qui revient très couramment mais pourtant n'est pas explicite. Cette erreur apparaît sous cette forme :

```
[fatal] rule MaRegle has non-LL(*) decision due
to recursive rule invocations reachable from alts
x,y. Resolve by left-factoring or using syntactic
predicates or using backtrack=true option.
```

Les nouvelles versions d'Xtext savent détecter cette erreur dans l'éditeur directement. Ce n'était pas le cas avec des versions plus anciennes, auquel cas ce paragraphe pourra

vous dépanner !

Cette erreur est levée par une expression comme celle-ci :

```
Expression :
    Expression '+' Expression |
    '(' Expression ')' |
    INT
;
```

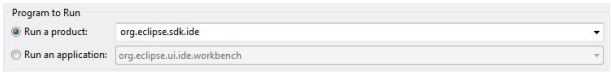
Pour corriger cette erreur, il suffit de factoriser la règle de cette manière :

```
Expression :
    TerminalExpression ('+' TerminalExpression)?
;
TerminalExpression :
    '(' Expression ')' |
    INT
;
```

4.4. Test de l'éditeur

4.4.1. Créer la configuration

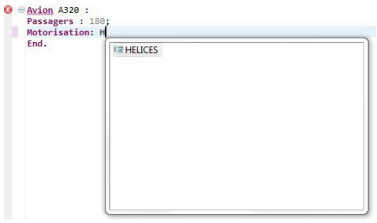
Une fois notre grammaire générée, il est temps de la tester au sein d'un éditeur. Pour cela, une méthode simple est de faire un clic droit sur le projet « *.ui » et de choisir « Run As > Launch Runtime Eclipse ». Cette option va lancer une copie exacte de votre instance Eclipse en cours d'exécution, ce qui peut entraîner un temps de chargement assez long pour de simples tests. L'autre solution consiste à définir soi-même une configuration au sein du panneau « Run configurations » d'Eclipse et de ne sélectionner que les plugins nécessaires à l'exécution du plugin « *.ui » (bouton « Add required plugins »). Dans l'onglet « Main », l'option choisie dans la section « Program to Run » est « Run a product : org.eclipse.sdk.ide »



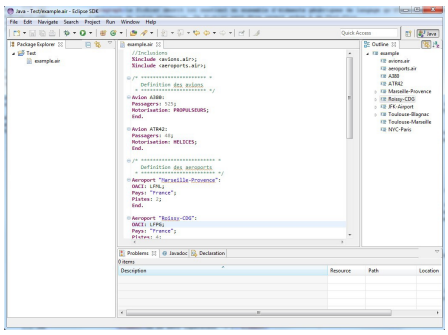
4.4.2. Testons notre grammaire !

Une fois notre instance de test lancée, il suffit de créer un nouveau projet vide « File > New > Project », et de créer à l'intérieur de ce projet un nouveau fichier « .air ». Xtext vous proposera alors de configurer votre projet pour être utilisé avec Xtext :

Vous pourrez donc profiter de l'autocomplétion et des autres outils associés pour créer vos fichiers.



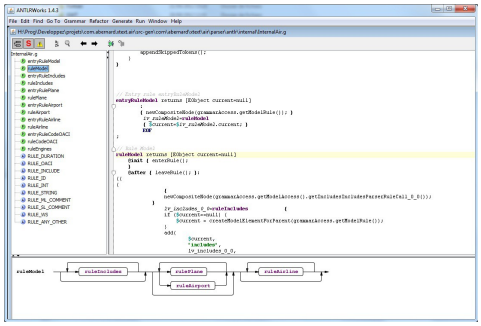
On peut d'ailleurs observer le résultat avec notre fichier d'exemple présenté en début de cet article. Xtext nous fournit bien un environnement convivial de travail, bien plus qu'un simple éditeur classique. Notez la présence de la Outline déjà configurée avec les attributs de la grammaire nommés « name » !



Notre IDE

5. Un outil bien pratique : ANTLRWorks

Il est parfois difficile de corriger des erreurs remontées lors de la génération de la grammaire, plus particulièrement celles exprimant un non-déterminisme de la grammaire (i.e. lorsque le parser ne peut pas « choisir » entre deux règles à suivre). L'outil ANTLRWorks permet de visualiser la grammaire sous forme de graphe et permet donc de corriger les erreurs plus facilement. ANTLRWorks est téléchargeable ici : Lien 153. Une fois lancé, il suffit d'ouvrir le fichier « .g » qui a été généré dans Eclipse. Il se trouve dans le package « src-gen/%votreprojet %.parser.antlr.internal ». On peut alors naviguer au sein des différentes règles de la grammaire et éventuellement voir au sein des graphes affichés si une règle comporte un non-déterminisme. Pour notre projet par exemple, on peut afficher la règle de base :



ANTLRWorks

L'intérêt est aussi de prendre un élément concret du langage et de pouvoir voir en quoi il est parsé.

6. Conclusion

Dans cet article nous aurons étudié les principales règles qui régissent la construction d'une grammaire sous Xtext. Nous avons cependant évoqué uniquement les plus grands principes et la richesse de la grammaire Xtext ne s'arrête pas là. La documentation officielle vous fournira éventuellement les réponses aux questions que vous pourriez encore vous poser.
Nous avons pu aussi constater qu'à partir d'un seul fichier de définition, le framework est capable de proposer un environnement de développement complet et convivial. En plus de proposer les outils nécessaires à l'exploitation efficace des fichiers d'un DSL, cet environnement est hautement personnalisable et ce sera le sujet d'un prochain article, focalisé sur la personnalisation de l'IDE.

Retrouvez l'article d'Alain Bernard en ligne : Lien 154

Android

Les dernières news

Android : 56 % des utilisateurs français n'ont aucune application payante, d'après une étude de Surikate, les bogues sont la principale raison de la désinstallation d'une app.

La société de marketing mobile Surikate vient de réaliser une nouvelle étude sur les utilisateurs d'Android en France.

Menée en partenariat avec l'Ifop, et avec plus de 9000 participants, cette enquête avance quelques chiffres intéressants – ou en tout cas surprenants. Deux qui retiendront un peu plus l'attention des développeurs.

Le premier révèle que la **principale raison de suppression** d'une application est la présence de **bogues (76 %)**. Le deuxième que **56 % des possesseurs d'appareils sous Android n'ont aucune application payante**. Ce qui confirme la prédominance du modèle Freemium (avec publicité ou monétisé avec des achats in-apps) sur la plateforme.

Autre constatation amusante, le lieu où **Google Play est le plus consulté** est… **dans le lit** (pour 55 % des utilisateurs).

Parmi les autres conclusions de l'étude on retiendra **ces quelques morceaux choisis** :

- 41 % des personnes sondées utilisent leur smartphone plus de quatre heures par jour ;
- 62 % des utilisateurs d'Android refusent de prêter leur smartphone mais 65 % de ceux qui possèdent des tablettes acceptent de les prêter ;
- 58 % utilisent au moins six applications par jour ;
- les jeux restent les applications privilégiées pour 71 % des utilisateurs ;
- 79 % des utilisateurs recommandent une application qui leur plait ;
- 86 % des personnes qui téléchargent des applications ont l'habitude d'en télécharger plusieurs pour les comparer entre elles et ne garder que la meilleure.

Vous pouvez découvrir l'intégralité des résultats en téléchargeant l'étude complète : Lien 155

Commentez cette news de Gordon Fowler en ligne : Lien 156

Les derniers tutoriels et articles

Accéder aux services Web via Android

Ce tutoriel a pour but de vous présenter l'utilisation des services Web sous Android.

1. Les services Web

Les services Web sont des services de traitement de la donnée exposée sur internet. Ils peuvent avoir plusieurs formes, provenir de plusieurs sites différents, faire des tâches diverses et être privés ou publics.
Exemples :

- données météorologiques ;
- calcul mathématique ;
- stockage d'informations ;
- etc.

Les services Web peuvent être codés en plusieurs langages (C#, Java, PHP, Ruby, Python, C, etc.).
Nous allons nous intéresser plus précisément aux services retournant des données en JSON (JavaScript Object Notation).
Pourquoi le JSON ? Eh bien pour une question de facilité, bien entendu vous pouvez utiliser le XML aussi, mais le JSON a la particularité d'être plus léger que XML, un gain de poids n'est pas anodin dans le domaine de la mobilité.

2. Le service Web de Test

Vous trouverez à ce lien le service Web de test que nous utiliserons par la suite : Lien 157
Ce service est développé en .NET sous forme de Web API.
Cette URL retourne une liste de points géographiques comportant les éléments suivants :

- Adress : adresse du lieu ;
- City : ville ;
- PostalCode : code postal ;
- Latitude : latitude ;
- Longitude : longitude.

3. Structure de notre application

Notre application va être structurée de la manière

suivante :

- vue (XML) ; ListView ;
- ArrayAdapter ;
- client pour le service Web ;
- modèle de données.

Nous allons aussi utiliser la bibliothèque GSON, qui permet de sérialiser/désérialiser en JSON, inutile de réinventer la roue.
Vous trouverez à ce lien le projet de la librairie GSON : Lien 158

4. Appel au service Web

Nous allons créer une classe pour effectuer nos appels.
Exemple : WebService.java
Nous retrouverons les éléments suivants dans notre classe :

- la base de l'URL pour le service Web ;
- l'objet Gson ;
- un constructeur ;
- une méthode private pour se connecter en HTTP au service Web et récupérer un InputStream ;
- des méthodes pour chaque API disponible.

```
public class WebService {

   private final String URL =
"http://testwebapi.lalimace.fr/api/Point";

   Gson gson;

   public WebService() {
       gson = new Gson();
   }

   private InputStream sendRequest(URL url)
throws Exception {

       try {
           // Ouverture de la connexion
           HttpURLConnection urlConnection =
(HttpURLConnection) url.openConnection();

           // Connexion à l'URL
           urlConnection.connect();

           // Si le serveur nous répond avec un
code OK
           if (urlConnection.getResponseCode() ==
HttpURLConnection.HTTP_OK) {
               return
urlConnection.getInputStream();
           }
       } catch (Exception e) {
           throw new Exception("");
       }
       return null;
   }

   public List<Point> getPoints() {

       try {
           // Envoi de la requête
           InputStream inputStream =
sendRequest(new URL(URL));

           // Vérification de l'inputStream
           if(inputStream != null) {
               // Lecture de l'inputStream dans
un reader
               InputStreamReader reader = new
InputStreamReader(inputStream);

               // Retourne la liste désérialisée
par le moteur GSON
               return gson.fromJson(reader, new
TypeToken<List<Point>>(){}.getType());
           }

       } catch (Exception e) {
           Log.e("WebService", "Impossible de
rapatrier les données :(");
       }
       return null;
   }
}
```

sendRequest sera notre méthode générique pour appeler via un objet HttpUrlConnection le service Web, ici on se contente de faire un simple appel de type GET sur une URL.
Le résultat retourné est toujours un InputStream que nous allons pouvoir traiter, nous pouvons avoir plusieurs types de données retournées par un service Web (XML, JSON, Image, Excel, etc.).
Le type InputStream est facilement manipulable avec le Reader qui va bien.
Notre cas se porte uniquement sur le JSON.
Nous avons aussi besoin d'un modèle, le flux étant de type JSON il faut désérialiser celui-ci pour le transformer en un objet exploitable facilement.
Nous allons donc créer un fichier Point.java, lequel contiendra les informations suivantes :

- les propriétés ;
- les accesseurs.

Vous pouvez aussi rajouter vos propres méthodes, comme toString etc.
Voici le code :

```
package com.aceart.formationwebservice.model;

public class Point {
   int ID;
   String Address;
   String City;
   String PostalCode;
   float Latitude;
   float Longitude;

   public int getID() {
       return ID;
   }
   public void setID(int iD) {
       ID = iD;
   }
   public String getAddress() {
       return Address;
   }
   public void setAddress(String address) {
       Address = address;
   }
   public String getCity() {
       return City;
   }
   public void setCity(String city) {
```

```
        City = city;
    }
    public String getPostalCode() {
        return PostalCode;
    }
    public void setPostalCode(String postalCode) {
        PostalCode = postalCode;
    }
    public float getLatitude() {
        return Latitude;
    }
    public void setLatitude(float latitude) {
        Latitude = latitude;
    }
    public float getLongitude() {
        return Longitude;
    }
    public void setLongitude(float longitude) {
        Longitude = longitude;
    }
}
```

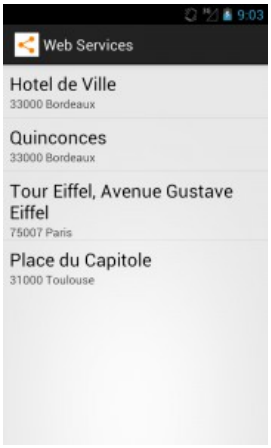
Avec la bibliothèque GSON tout se fait en une ligne pour transformer l'inputStream en Objet.

```
return gson.fromJson(reader, new
TypeToken<List<Point>>(){}.getType());
```

À partir de là vous obtenez donc une liste d'objets provenant d'un service Web retournant du JSON.

5. Conclusion

Vous trouverez les sources de la démo en bas de l'article, contenant la partie affichage avec l'ArrayAdapter et la ListView, je ne vais pas m'attarder sur ces deux points car nous les avons déjà vus dans d'autres cours.



Dans un prochain tutoriel nous aborderons l'utilisation du Spring Framework qui permet d'utiliser des services Web de type REST facilement.

6. Annexes

Source : FormationWebservice (Lien 159)

Retrouvez l'article de Pierre-Emmanuel Mercier en ligne : Lien 160

Styles & Thèmes - Comment utiliser les thèmes et les styles.

On est souvent amené à vouloir customiser les bouts d'interface proposés par défaut par le système (les listes, les Tabs, la barre d'application). Une des méthodes est de dupliquer la ressource et de l'adapter à nécessité... L'autre méthode, plus appropriée, consiste à utiliser les thèmes pour adapter le système à son interface.

1. Les Styles

Cet article vient en complément de l'introduction des styles et des thèmes par Cyril Mottier (que vous pouvez lire ici : Lien 161), toutefois, nous allons repasser brièvement sur les concepts de base.

1.1. Définition

Les « styles » permettent de transformer des éléments structurés (HTML, XML) simples et pas forcément très lisibles en une version formatée et affichable. On pense bien entendu aux « Cascading Style Sheet » (CSS) du langage HTML (SGML) qui remontent aux années 1980.
Le style permet donc de définir un certain nombre d'attributs (cosmétiques ou non) qui seront les valeurs par défaut des éléments auxquels le style est appliqué.
L'intérêt est double : d'une part permettre d'automatiquement fournir des attributs cosmétiques à des éléments, mais surtout de maintenir, à un seul endroit, ce qu'on appelle le « Look'n Feel », c'est-à-dire la marque visuelle du site (ou de l'application).

1.2. Dans Android

1.2.1. Petit rappel sur les ressources

Les ressources sont stockées d'une manière un peu spéciale dans Android. Toutes les ressources partagent le même paradigme d'accès :
en Java, les ressources sont accessibles par l'utilisation de R.xxxx.nnnn, où xxxx est le type de ressource ('layout', 'id', 'string', etc. et bien entendu 'style'), et nnnn son « nom ». Les ressources prédéfinies d'Android sont aussi accessibles en Java par android .R.xxxx.nnnn. Dans tous les cas, ce sont des valeurs entières (int) générées à la compilation ;
dans les fichiers XML les ressources sont référencées par l'utilisation de '@xxxx/nnnn' (ou de '@android :xxxx/nnnn' pour les ressources natives à Android) ;
ainsi, une ressource de type « string » sera référencée en Java par : R.string.LeNomDeMaString et en XML : @string/LeNomDeMaString.

1.2.2. Utilisation des Styles

Au contraire de HTML (où chaque 'classe' a son propre style et chaque élément peut avoir plusieurs classes, et donc plusieurs styles), sous Android, chaque élément

(View) ne peut avoir qu'un seul et unique style.
Le choix du style est fait au moment de la construction de l'élément, ou bien de manière explicite (un attribut 'android:style' a été spécifiquement passé à la construction de l'élément) ou bien de manière implicite (le style par défaut de l'élément est utilisé).

```
<LinearLayout>
        <TextView id='id1'
android:style='@style/MyStyle'/>
        <TextView id='id2'/>
</LinearLayout>
```

Dans l'exemple ci-dessus, on a deux éléments "TextView", l'un (id1) aura un style explicitement défini, l'autre (id2) aura le style « par défaut » des TextViews. Nous verrons plus loin comment ce style par défaut est défini.
Il est bien sûr possible d'utiliser directement les styles Android. « @android:style/ListView » par exemple sera le style par défaut appliqué aux éléments ListView.

1.2.3. Définir son propre style

Les styles sont des ressources de type « style ». Et donc, référencés en XML par « @style ». Il suffit donc de créer un fichier XML (dans le répertoire 'values' des ressources) contenant un ou plusieurs styles. Par exemple :

```
<resource>
  <color name="light_grey">#e6eced</color>
  <style name="MyStyle">
    <item name="android:background">
      @color/light_grey
    </item>
  </style>
</resource>
```

Ici, le style "MyStyle" a été défini et va appliquer automatiquement une couleur de background à tous les éléments qui l'utiliseront.

1.2.4. Définition en Cascade

Il est possible d'utiliser un « parent » pour le style. Tous les attributs du style parent seront inclus (et éventuellement surchargés par le style en cours).
Le parent peut être « implicite » en utilisant le point (.) pour spécifier le nom du style :

```
<resource>
  <style name="Layout">
    <item name="android:background">
      @color/light_grey
    </item>
    <item name="android:padding">5sp</item>
  </style>
  <style name="Layout.Bar">
    <item name="android:background">
      @color/dark_grey
    </item>
  </style>
<resource>
```

Ici, le style « Layout.Bar » aura comme parent le style « Layout » (et donc avec un attribut android:padding the « 5 sp »).
Il peut aussi être explicite en utilisant l'attribut « parent » :

```
<resource>
  <style name="MyButton"
parent="@android:style/Widget.Button">
    <item name="android:background">
      @drawable/yellow_button
    </item>
    <item name="android:textAppearance">
      @style/TextAppearance.Label
    </item>
    <item name="android:textColor">
      @color/red
    </item>
    <item name="android:minWidth">80dp</item>
  </style>
</resource>
```

Ici le style "MyButton" aura comme parent le style prédéfini « Widget.Button » d'Android, mais une largeur minimale de 80 dp etc.

1.2.5. Que peut-on « styliser » ?

Tout d'abord qui peut recevoir un style et quel attribut peut-on lui appliquer ? Tous les éléments d'interface (View) peuvent recevoir un style. Quant aux attributs « stylisables », vous trouverez la liste exhaustive dans android.R.stylable : Lien 162.
Voici quelques attributs cosmétiques « classiques » :

- minWidth, minHeight : très utiles pour les widgets ;
- padding : correspond à la taille additionnelle autour de l'élément ;
- background : le « drawable » à dessiner en fond (généralement une couleur, ou un dégradé) ;
- textColor, textSize, textStyle et typeface (TextView) : définissent la manière donc le texte sera écrit ;
- margin (ViewGroup) : espace à l'intérieur de l'élément, qui ne sera pas utilisé par les éléments fils.

Quant aux styles desquels vous pouvez hériter (que vous pouvez utiliser comme parents), ce sont des ressources de type style donc vous trouverez votre bonheur dans android.R.style : Lien 163.

1.2.6. Le cas particulier de textAppearance

Pour un certain nombre d'éléments, il est possible de spécifier un second style : textAppearance. Cet attribut fait référence à un style, mais seuls les attributs du style concernant le texte seront utilisés, c'est-à-dire : textColor (mais aussi textColorLink, textColorHighlight.), textSize, textStyle, typeface, et textAllCaps.
Cela évite par exemple d'avoir quatre styles différents de « boutons » en fonction de la taille du texte... Il y a un seul et unique style de bouton (@android:style/Widget.Button) et quatre styles différents de texte ('normal', 'Small', 'Medium' et 'Large') : les « TextAppearance.Small », etc.
Comme nous le verrons plus tard, cette séparation est d'autant plus pratique qu'il est possible de « modifier » les TextAppearance.Small.

1.3. Conclusion sur les styles

Normalement, à ce point, vous devriez pouvoir assigner un

style à vos éléments d'interface, et ainsi vous éviter de modifier tous vos éléments quand vous décidez de changer la fonte de votre application.

2. Les thèmes

Vous devez tout de même vous dire que décidément, placer l'attribut 'style' à tous vos éléments va être très long, ennuyeux, et pas du tout pratique, et je vous l'accorde. Il existe néanmoins un outil pour vous éviter ce long travail : les thèmes.

2.1. Utilisation

Un thème n'est rien de plus qu'une liste de valeurs d'attributs (un peu comme les styles au demeurant). La liste de ces attributs (de thème) est accessible dans android.R.attr : Lien 164. Mais nous verrons plus loin qu'on peut y rajouter des attributs propres à l'application.
Pour des raisons de simplicité (mais pas forcément de lisibilité et/ou compréhension), Google a décidé de représenter les thèmes comme des ressources de type « style » qui doivent avoir comme parent le style « Theme » d'une manière ou d'une autre (directement ou non donc). Ainsi, si vous déclarez un style « Toto » dont le parent est Theme.Holo.Dark, vous aurez déclaré un thème !
Chaque activité reçoit un thème unique. Si une activité n'a pas de thème spécifique, le thème est celui par défaut de l'application.
Ce thème est utilisé pendant la construction de l'interface (inflater, setContentView) pour récupérer certains attributs par défaut (y compris le style). Ceci est aussi vrai pour les « layouts » système. Plutôt que de définir votre propre R.layout. list_item pour utiliser vos attributs cosmétiques, vous pourrez utiliser celui d'Android (android.R.layout.single_list_item), qui sera customisé en fonction du thème !
Les valeurs des attributs du thème sont accessibles depuis tout fichier XML par l'utilisation de « ? attr/nom_de_lattribut » ou « ? android:attr/nom_de_lattribut » s'il est natif à Android.
Ainsi,

```
<LinearLayout>
        <TextView id='id1'
android :textAppearance=' ?
android:attr/textAppearanceSmall'/>
</LinearLayout>
```

va indiquer que l'attribut « textAppearanceSmall » du thème en cours doit être utilisé. Dans le cas d'un thème « Holo.Dark », le texte sera blanc, dans le cas de « Holo.Light », il sera donc noir.
Il y a bien sûr de fortes chances pour que la valeur de cet attribut soit simplement « @android:style/TextAppearance.Small ».

2.2. Créer un thème

Comme nous l'avons vu, les thèmes sont définis exactement comme les styles. Mais pour vous aider à vous y retrouver, je vous conseille vivement de séparer vos fichiers de thème et vos fichiers de styles.

2.2.1. Surcharger un thème existant

La liste des attributs est assez volumineuse, mais comme votre thème aura forcément un thème comme parent, tous les attributs non définis seront ceux du parent.

```
<resource>
  <style name="MyTheme"
parent="@android:style/Theme.Holo.Light">
    <item name="android:colorBackground">
      @color/light_grey
    </item>
    <item
name="android:colorBackgroundCacheHint">
      @color/light_grey
    </item>
  </style>
</resource>
```

Une fois que vous avez défini ce thème, vous devriez le voir apparaitre dans l'éditeur d'interface. Par défaut le background des éléments est désormais la couleur light_grey. Et les ListView utiliseront cette couleur pour le « fading ». Et pourtant sans rien toucher à aucun attribut des layouts !
À noter qu'à partir d'Android 3.0, le thème « Theme.Holo » est à privilégier.

2.2.2. Surcharger les styles par défaut

Dans la liste des attributs des thèmes, vous trouverez un certain nombre d'entrées intéressantes, en particulier sur les styles par défaut. Par exemple, l'attribut : buttonStyle. Cet attribut définit le style par défaut des boutons.
Pour le surcharger, commencez par créer votre propre style de bouton :

```
<resource>
  <style name="MyButton"
parent="@android:style/Widget.Button">
    <item name="android:background">
      @drawable/MyButton
    </item>
  </style>
</resource>
```

Puis surchargez l'attribut buttonStyle dans le thème :

```
<resource>
  <style name="MyTheme"
parent="@android:style/Theme.Holo">
    <item name="android:buttonStyle">
      @style/MyButton
    </item>
  </style>
</resource>
```

Et voilà. Tous vos boutons auront désormais le style « MyButton » par défaut. Et, plus intéressant encore, tout layout (système) incluant un bouton aura ce style !
À noter qu'une fois le thème sélectionné dans l'éditeur d'interface, vous verrez aussi directement l'aspect des boutons dans la sélection à gauche.

2.2.3. Créer ses propres attributs

Bien sûr, il serait utile de pouvoir rajouter des attributs spécifiques à votre application. Là encore c'est possible.

En utilisant la ressource « attr »

```
<resource>
  <attr name="generalMargin" format="reference|
dimension"/>
  <style name="MyTheme"
parent="@android:style/Theme.Holo">
    <item name="generalMargin">5sp</item>
  </style>
</resource>
```

Et bien évidemment, vous pourrez utiliser votre attribut avec « ?attr/generalMargin »

```
<resource>
  <style name="MyButton"
parent="@android:style/Widget.Button">
    <item name="android:padding">
      ?attr/generalMargin
    </item>
  </style>
</resource>
```

2.3. Spécifier le thème pour votre activité

Bien entendu, l'éditeur d'interface d'Eclipse vous permet de choisir le thème pour afficher tel ou tel layout, mais il vous faudra éditer le 'manifest' (là où est déclarée votre activité) pour lui attribuer un thème. C'est d'ailleurs comme cela qu'on fait une véritable boite de dialogue.
Bien entendu, si l'ensemble de votre application n'utilise qu'un seul thème, libre à vous de l'utiliser comme thème par défaut au niveau de l'application.
Il est possible de modifier le thème depuis Java, en utilisant la ressource ad hoc R.style.MyTheme : setTheme() dans l'activité. À noter que le thème ne modifie pas l'UI en cours, il faut donc reconstruire la vue, et appeler la fonction avant le setContentView. C'est la seule méthode pour proposer plusieurs thèmes à l'utilisateur dans la même application.
Prenons un exemple, j'ai une activité qui peut s'afficher sous forme de dialogue ('PICK'), ou 'normalement' ('VIEW').
J'ai donc deux thèmes : « MyTheme » qui hérite de « @android :style/Theme.Holo » et « MyTheme.Dialog » qui hérite de « @android :style/Theme.Holo.Dialog ». Il suffit dans mon activité que j'écrive :

```
@Override
public void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState); // <=
toujours en premier
        boolean isDialogCall =
getIntent().getName().equals(Intent.ACTION_PICK);
        setTheme(isDialogCall ?
R.style.MyTheme.Dialog : R.style.MyTheme);
        setContentView(R.layout.my_layout); // <=
maintenant on peut créer les views
}
```

À noter qu'il n'est pas nécessaire d'avoir surchargé le thème pour le faire, on aurait très bien pu utiliser android.R.style.Holo_Dark et android.R.style.Holo_Dark_Dialog par exemple.

3. Un Exemple

Dans cet exemple, nous allons partir d'une application très simple, incluant des boutons, des TextView, des CheckBox et même une ListView qui utilise les layouts par défaut. Et nous allons la customiser pour avoir 'notre' look'n feel (de très mauvais goût ici, mais c'est juste pour les besoins de la démonstration), sans toucher un pouce du fichier de layout initial.
D'abord le fichier main.xml du layout principal de l'activité. Le code Java ne fait que remplir la List-View avec des données.

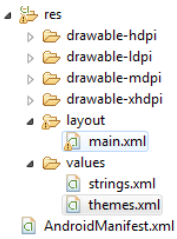
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >
  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
    <Button android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Button" />
    <Button  android:id="@+id/button2"
      style="?android:attr/buttonStyleSmall"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Button" />
    <ToggleButton android:id="@+id/toggleButton1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="ToggleButton" />
  </LinearLayout>
  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
    <CheckBox android:id="@+id/checkBox1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="CheckBox" />
    <RadioButton android:id="@+id/radioButton1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="RadioButton" />
  </LinearLayout>
  <ListView android:id="@+id/listView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
      <!-- Preview:
listitem=@android:layout/simple_list_item_checked
-->
  </ListView>
</LinearLayout>
```

Comme vous pouvez le voir, aucune customisation d'interface, rien que des éléments par défaut sans modification cosmétique. Nous n'y toucherons plus.
Au passage, vous noterez que le « Small Button » indique clairement aller chercher son style dans le thème.

Voici le résultat :

3.1. Tout commence par un thème

Créer le thème dans un fichier « themes.xml » :



Le contenu est simple :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <style name="ThemeTest"
parent="@android:style/Theme.Holo">
  </style>
</resources>
```

Une fois le fichier enregistré, vous pourrez voir le thème apparaitre dans l'éditeur de layouts, mais avant, nous allons modifier le 'manifest' de notre application pour que l'activité utilise ce thème par défaut.
Nous allons donc rajouter l'attribut suivant dans la déclaration de l'activité :
android:theme="@style/ThemeTest"
Pour l'instant, rien n'a changé, puisque nous héritons simplement de 'Theme.Holo'.

3.2. Réglages de base

Ici, nous allons commencer par modifier le 'background', et la couleur du texte.
Pour le background, déjà, première complication, il nous faut passer par un nom de couleur. Qu'à cela ne tienne, on définit une ressource de type « color » (dans le même fichier), et on y fait référence avec @color/le_nom_de_la_couleur.
Il en est de même pour la couleur du texte.
Voici le fichier themes.xml après modification et le résultat :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="dark_green">#002000</color>
  <color name="yellow">#ffff00</color>
  <style name="ThemeTest"
parent="@android:style/Theme.Holo">
    <item name="android:background">
      @color/dark_green
    </item>
    <item name="android:textColor">
      @color/yellow
    </item>
  </style>
</resources>
```

Premiers constats : la couleur du fond a bien été changée partout, mais la couleur de texte n'a été appliquée que sur les TextViews ! Voici pourquoi.
Si on regarde le style Widget.Holo.Button (le style par défaut des boutons), celui-ci fait directement référence à une couleur : @android:color/primary_text_holo_dark, qui est en fait un sélecteur de couleur (un peu comme un sélecteur de background) dont je vous mets le code ici :

```
<selector
xmlns:android="http://schemas.android.com/apk/res
/android">
  <item android:state_enabled="false"
android:color="@android:color/bright_foreground_d
isabled_holo_dark"/>
  <item android:state_window_focused="false"
android:color="@android:color/bright_foreground_h
olo_dark"/>
  <item android:state_pressed="true"
android:color="@android:color/bright_foreground_h
olo_dark"/>
  <item android:state_selected="true"
android:color="@android:color/bright_foreground_h
olo_dark"/>
  <item android:state_activated="true"
android:color="@android:color/bright_foreground_h
olo_dark"/>
  <item
android:color="@android:color/bright_foreground_h
olo_dark"/>
  <!-- not selected -->
</selector>
```

Ce qui veut dire, qu'en fonction de l'état du bouton, la couleur ne sera pas la même, et que le style du bouton va définir en dur la couleur du texte !

3.3. Surcharger les Styles

Il va donc falloir que nous surchargions, les styles de boutons. Pour les faire pointer sur notre couleur.
On commence par créer le style de bouton (on va en profiter pour faire de même avec les CheckBoxes et radio-

buttons). Voici donc le nouveau fichier de thème :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="dark_green">#002000</color>
  <color name="yellow">#ffff00</color>
  <style name="Widget.ThemeTest.Button"
parent="@android:style/Widget.Holo.Button">
    <item name="android:textColor">
      ?android:attr/textColor
    </item>
  </style>
  <style name="Widget.ThemeTest.Button.Small"
parent="@android:style/Widget.Holo.Button.Small">
    <item name="android:textColor">
      ?android:attr/textColor
    </item>
  </style>
  <style name="Widget.ThemeTest.Button.Toggle"
parent="@android:style/Widget.Holo.Button.Toggle"
>
    <item name="android:textColor">
      ?android:attr/textColor
    </item>
  </style>
  <style
name="Widget.ThemeTest.CompoundButton.CheckBox"
parent="@android:style/Widget.Holo.CompoundButton
.CheckBox">
    <item name="android:textColor">
      ?android:attr/textColor
    </item>
  </style>
  <style
name="Widget.ThemeTest.CompoundButton.RadioButton
"
parent="@android:style/Widget.Holo.CompoundButton
.RadioButton">
    <item name="android:textColor">
      ?android:attr/textColor
    </item>
  </style>
  <style name="ThemeTest"
parent="@android:style/Theme.Holo">
    <item name="android:background">
      @color/dark_green
    </item>
    <item name="android:textColor">
      @color/yellow
    </item>
    <item name="android:buttonStyle">
      @style/Widget.ThemeTest.Button
    </item>
    <item name="android:buttonStyleSmall">
      @style/Widget.ThemeTest.Button.Small
    </item>
    <item name="android:buttonStyleToggle">
      @style/Widget.ThemeTest.Button.Toggle
    </item>
    <item name="android:checkboxStyle">
  @style/Widget.ThemeTest.CompoundButton.CheckBox
    </item>
    <item name="android:radioButtonStyle">
@style/Widget.ThemeTest.CompoundButton.RadioButto
n
    </item>
  </style>
</resources>
```

Et le résultat :

Finalement, nous avons bien la bonne couleur partout, et il suffit de la changer dans le thème (textColor) !
On pourrait dire : tant de code que ça dans les thèmes et styles, juste pour changer les couleurs ? Oui sans doute, pour une seule fenêtre dans une application de test, il vaut probablement mieux utiliser les attributs de style directement. Mais si vous avez une dizaine de layouts, et que vous utilisez ceux du système (ListView ?), le passage par le thème devient bien vite nécessaire (et vous évite de réinventer la roue dans bien des cas).

4. Conclusion

Grâce aux thèmes, vous pouvez, depuis un unique endroit modifier l'aspect visuel de votre application. Mieux encore, les ressources pouvant être sélectionnées en fonction du device, de la langue, etc., il est possible d'avoir des thèmes variables en fonction de la taille de l'écran, de son orientation.
Le plus intéressant encore une fois, est de voir, d'un coup, l'ensemble des layouts système adopter votre « Look'n Feel ».
Pour vous simplifier la vie, je vous conseille de suivre ces quelques règles :

- mis à part les attributs de structuration des layouts (gravité, width/height.), tout attribut d'un élément d'interface doit être défini par le style ;
- dès qu'une valeur est utilisée par plusieurs styles, la définir comme attribut dans le thème ;
- conserver au maximum les styles par défaut (et les surcharger depuis le thème), ceci permet d'utiliser les layouts système de manière transparente et avec le même Look'n Feel.

Si vous comptez fournir plusieurs « thèmes » à votre application, je vous conseille aussi :

- de ne jamais pointer directement sur un style, mais sur un attribut du thème qui définira le style. Vous pourrez ainsi, en fonction du thème, modifier le style des éléments.

Retrouvez l'article de Nicolas Romantzoff en ligne : Lien 165

Liens

Lien 01 : http://support.microsoft.com/kb/450913/fr
Lien 02 : http://office.microsoft.com/fr-ca/word-help/presentation-des-nouvelles-extensions-de-noms-de-fichier-et-des-formats-xml-ouverts-microsoft-office-HA010006935.aspx
Lien 03 : http://didier-gonard.developpez.com/tutoriels/office/vba-qu-est-que-c-est/
Lien 04 : http://www.inserthtml.com/
Lien 05 : http://www.inserthtml.com/2012/06/css-filters/
Lien 06 : http://inserthtml.developpez.com/tutoriels/css/filtres-css3-html-images/
Lien 07 : http://www.paulund.co.uk/
Lien 08 : http://www.paulund.co.uk/use-box-model-for-easier-widths
Lien 09 : http://caniuse.com/css3-boxsizing
Lien 10 : http://paulund.developpez.com/tutoriels/css/box-sizing/
Lien 11 : http://www.paulund.co.uk/how-to-create-a-css-slide-out-fixed-navigation
Lien 12 : http://paulund.developpez.com/tutoriels/css/menu-fixe-slide/fichiers/menu_fixe_slide.html
Lien 13 : http://paulund.developpez.com/tutoriels/css/menu-fixe-slide/
Lien 14 : http://torgar.developpez.com/articles/css/transitions-css3/fichiers/exemple-1.html
Lien 15 : http://www.w3.org/TR/css3-transitions/#properties-from-css-
Lien 16 : http://torgar.developpez.com/articles/css/transitions-css3/fichiers/exemple-2.html
Lien 17 : http://jqueryui.com/demos/effect/easing.html
Lien 18 : http://matthewlein.com/ceaser/
Lien 19 : http://torgar.developpez.com/articles/css/transitions-css3/fichiers/exemple-3.html
Lien 20 : http://torgar.developpez.com/articles/css/transitions-css3/fichiers/exemple-4.html
Lien 21 : http://torgar.developpez.com/articles/css/transitions-css3/fichiers/exemple-5.html
Lien 22 : http://torgar.developpez.com/articles/css/transitions-css3/
Lien 23 : http://www.paulund.co.uk/understanding-the-viewport-meta-tag
Lien 24 : http://paulund.developpez.com/tutoriels/html/metatag-viewport/
Lien 25 : http://www.paulund.co.uk/add-telephone-number-links-with-html5
Lien 26 : http://paulund.developpez.com/tutoriels/html/liens-telephone/
Lien 27 : http://www.html5rocks.com/en/tutorials/pagevisibility/intro/
Lien 28 : http://www.html5rocks.com/en/
Lien 29 : http://www.w3.org/TR/page-visibility/
Lien 30 : https://developers.google.com/chrome/whitepapers/prerender
Lien 31 : http://www.html5rocks.com/en/tutorials/notifications/quick/
Lien 32 : https://developers.google.com/chrome/whitepapers/prerender
Lien 33 : http://www.html5rocks.com/features/performance
Lien 34 : http://www.w3.org/2010/webperf/
Lien 35 : http://ie.microsoft.com/testdrive/Performance/PageVisibility/Default.html
Lien 36 : http://thechangelog.com/post/29416591924/visibility-js-a-wrapper-for-the-page-visibility-api
Lien 37 : http://coderwall.com/p/zuydwa
Lien 38 : http://dmouronval.developpez.com/tutoriels/javascript/utiliser-api-pagevisibility/
Lien 39 : http://www.paulund.co.uk/javascript-full-screen-api
Lien 40 : http://caniuse.com/fullscreen
Lien 41 : http://paulund.developpez.com/tutoriels/css/pseudo-class-invalid/
Lien 42 : https://github.com/sindresorhus/screenfull.js/
Lien 43 : http://paulund.developpez.com/tutoriels/jquery/full-screen-api/fichiers/demo.html
Lien 44 : http://paulund.developpez.com/tutoriels/jquery/full-screen-api/
Lien 45 : http://qt.developpez.com/actu/38218/Le-Qt-Project-est-la-le-projet-d-open-gouvernance-pour-le-framework-C-est-arrive-a-terme/
Lien 46 : http://www.developpez.net/forums/d1278682/c-cpp/bibliotheques/qt/direct-qt-devdays-2012-a/
Lien 47 : http://www.developpez.net/forums/d1278983/c-cpp/bibliotheques/qt/plateformes/android-necessitas-port-android-qt-integre-qt-project/
Lien 48 : http://www.developpez.net/forums/d1209764/c-cpp/bibliotheques/qt/sortie-qt-5-beta-2-a/#post6978459
Lien 49 : http://www.developpez.net/forums/d1278682/c-cpp/bibliotheques/qt/direct-qt-devdays-2012-a/
Lien 50 : http://qt-project.org/wiki/Qt500beta2KnoLien
Lien 51 : http://qt-project.org/downloads
Lien 52 : http://www.developpez.net/forums/d1209764/c-cpp/bibliotheques/qt/sortie-qt-5-a/
Lien 53 : http://www.developpez.net/forums/d1276890/c-cpp/cpp/microsoft-renforce-support-cpp-11-visual-studio-2012-a/
Lien 54 : http://www.developpez.net/forums/d1276206/c-cpp/cpp/futur-cpp/
Lien 55 : http://www.boost.org/doc/libs/1_51_0/doc/html/proto.html
Lien 56 : http://en.wikipedia.org/wiki/Extended_Backus-Naur_Form
Lien 57 : http://boost-spirit.com/
Lien 58 : http://fr.wikipedia.org/wiki/Lex_et_yacc
Lien 59 : http://blitz.sourceforge.net/
Lien 60 : http://en.wikipedia.org/wiki/Loop_fusion
Lien 61 : http://www.boost.org/libs/proto
Lien 62 : http://en.wikipedia.org/wiki/Template_metaprogramming
Lien 63 : http://www.boost.org/
Lien 64 : http://en.wikipedia.org/wiki/Macro_%28computer_science%29#Lisp_macros
Lien 65 : http://www.imdb.com/video/screenplay/vi1032782617/
Lien 66 : http://cpp.developpez.com/redaction/data/pages/rubrique/cpp/boost/proto/proto1/
Lien 67 : http://en.wikipedia.org/wiki/Functional_programming
Lien 68 : http://en.wikipedia.org/wiki/Lambda_calculus
Lien 69 : http://en.wikipedia.org/wiki/Formal_system
Lien 70 : http://en.wikipedia.org/wiki/Lisp_%28programming_language%29
Lien 71 : http://altdevblogaday.com/2011/11/22/parallel-implementations
Lien 72 : http://cpp.developpez.com/carmack/analyse-statique
Lien 73 : http://en.wikipedia.org/wiki/Big_ball_of_mud
Lien 74 : http://dlang.org/function.html
Lien 75 : http://cpp.developpez.com/faq/cpp/?page=divers#OPTIMISATION_NRVO
Lien 76 : http://en.wikipedia.org/wiki/Single_assignment#Single_assignment

Lien 77 : http://cpp.developpez.com/redaction/data/pages/rubrique/cpp/carmack/fonctionnel/
Lien 78 : http://jeux.developpez.com/telecharger/detail/id/2324/RPG-Maker-VX
Lien 79 : http://fr.wikipedia.org/wiki/MUGEN
Lien 80 : http://jeux.developpez.com/telecharger/detail/id/2325/Adventure-Game-Studio
Lien 81 : http://jeux.developpez.com/telecharger/detail/id/2327/3D-Game-Studio
Lien 82 : http://jeux.developpez.com/telecharger/detail/id/2328/GameMaker
Lien 83 : http://jeux.developpez.com/telecharger/detail/id/2669/Unreal-Development-Kit-UDK
Lien 84 : http://jeux.developpez.com/telecharger/detail/id/2736/CryEngine-3-SDK
Lien 85 : http://jeux.developpez.com/telecharger/detail/id/2670/Unity
Lien 86 : http://jeux.developpez.com/telecharger/detail/id/2755/NeoAxis
Lien 87 : http://python.developpez.com/telecharger/detail/id/1957/Pygame
Lien 88 : http://jeux.developpez.com/telecharger/index/categorie/557/Bibliotheques
Lien 89 : http://jeux.developpez.com/tutoriels/?page=prog2d#sdl
Lien 90 : http://jeux.developpez.com/telecharger/index/categorie/557/Bibliotheques
Lien 91 : http://sgbd.developpez.com/
Lien 92 : http://flash.developpez.com/telecharger/index/categorie/276/EDI-et-editeurs-ActionScript
Lien 93 : http://sii-rennes.developpez.com/articles/un-chat-en-html5-avec-les-websockets/
Lien 94 : http://browserquest.mozilla.org/
Lien 95 : http://www.scirra.com/
Lien 96 : http://developers.facebook.com/
Lien 97 : http://jeux.developpez.com/telecharger/index/categorie/154/Outils-Jeux
Lien 98 : http://www.developpez.com/telecharger/index/categorie/449/UML
Lien 99 : http://www.developpez.net/forums/f144/applications/developpement-2d-3d-jeux/projets/
Lien 100 : http://ios.developpez.com/cours/
Lien 101 : http://a-renouard.developpez.com/tutoriels/ios/installation-developer-tools/
Lien 102 : http://android.developpez.com/cours/
Lien 103 : http://sourceforge.net/projects/minpspw/
Lien 104 : http://www.playstation.com/pss/developer/index_e.html
Lien 105 : http://krachik.developpez.com/tutoriels/pascal/install-config-fpc4nds/
Lien 106 : http://ds-homebrew.over-blog.com/
Lien 107 : http://gfx.developpez.com/prog-gba/
Lien 108 : http://www.codemii.com/2008/08/10/tutorial-1-setting-up-the-environment/
Lien 109 : http://jeux.developpez.com/tutoriels/?page=directx#xna
Lien 110 : http://msdn.microsoft.com/en-us/library/bb203937.aspx
Lien 111 : http://www.microsoft.com/en-us/kinectforwindows/develop/
Lien 112 : http://jeux.developpez.com/
Lien 113 : http://www.developpez.net/forums/f66/applications/developpement-2d-3d-jeux/
Lien 114 : http://jeux.developpez.com/telecharger/index/categorie/154/Outils-Jeux
Lien 115 : http://jeux.developpez.com/telecharger/index/categorie/557/Bibliotheques
Lien 116 : http://jeux.developpez.com/tutoriels/jeux-video/
Lien 117 : http://djibril.developpez.com/tutoriels/perl/installation-modules/
Lien 118 : http://www.php.net/manual/fr/function.session-save-path.php
Lien 119 : http://www.php.net/manual/fr/function.session-name.php
Lien 120 : http://www.php.net/manual/fr/function.session-start.php
Lien 121 : http://php.net/manual/fr/function.session-write-close.php
Lien 122 : http://search.cpan.org/dist/PHP-Session/
Lien 123 : http://search.cpan.org/search?query=English&mode=module
Lien 124 : http://search.cpan.org/search?query=perlvar&mode=all
Lien 125 : http://search.cpan.org/dist/CGI-Session/
Lien 126 : http://apache.developpez.com/cours/
Lien 127 : http://php.developpez.com/cours/
Lien 128 : http://perl.developpez.com/faq/perl/?page=sectionA32
Lien 129 : http://fr2.php.net/manual/fr/book.session.php
Lien 130 : http://search.cpan.org/dist/CGI/
Lien 131 : http://search.cpan.org/search?query=English&mode=all
Lien 132 : http://djibril.developpez.com/tutoriels/perl/sessions-php/fichiers/sessions.zip
Lien 133 : http://djibril.developpez.com/tutoriels/perl/sessions-php/fichiers/cgi-bin.zip
Lien 134 : http://djibril.developpez.com/tutoriels/perl/sessions-php/
Lien 135 : http://armel-ndjobo.developpez.com/memocertifjava6/
Lien 136 : http://armel-ndjobo.developpez.com/tutoriels/java/chap1-declaration-et-controle-acces/#LIX-B
Lien 137 : http://armel-ndjobo.developpez.com/tutoriels/java/chap1-declaration-et-controle-acces/
Lien 138 : http://code.google.com/p/google-web-toolkit/downloads/list
Lien 139 : http://www.developpez.net/forums/d1275785/webmasters-developpement-web/javascript/google-sort-web-toolkit-2-5-a/
Lien 140 : http://www.playframework.org/
Lien 141 : https://blogs.oracle.com/geertjan/entry/welcome_to_me
Lien 142 : http://java.net/projects/nbplay
Lien 143 : https://blogs.oracle.com/geertjan/entry/play_framework_plugin_for_netbeans
Lien 144 : https://blogs.oracle.com/geertjan/entry/play_framework_plugin_for_netbeans1
Lien 145 : http://www.developpez.net/forums/d1276499/java/developpement-web-java/frameworks/autres/play/support-framework-web-play-sous-netbeans-plugin-nbplay/
Lien 146 : http://gkemayo.developpez.com/eclipse/intro-Xtext/
Lien 147 : http://eclipse.developpez.com/cours/?page=platform-cat#plugin-dev
Lien 148 : http://fr.wikipedia.org/wiki/Grammaire_formelle
Lien 149 : http://fr.wikipedia.org/wiki/Domain-specific_programming_language
Lien 150 : http://www.eclipse.org/Xtext/download.html
Lien 151 : http://download.eclipse.org/releases/juno
Lien 152 : http://www.eclipse.org/Xtext/documentation.html#grammarLanguage
Lien 153 : http://www.antlr.org/download.html
Lien 154 : http://alain-bernard.developpez.com/tutoriels/eclipse/creation-grammaire-xtext/
Lien 155 : http://fr.slideshare.net/cedricbuisson/etude-surikateifop-sur-le-comportement-des-utilisateurs-dandroid
Lien 156 : http://www.developpez.net/forums/d1280232/java/general-java/java-mobiles/android/android-56-utilisateurs-francais-n-ont-aucune-application-payante/

Lien 157 : http://testwebapi.lalimace.fr/api/Point
Lien 158 : http://code.google.com/p/google-gson/
Lien 159 : http://www.ace-art.fr/wordpress/wp-content/uploads/2012/10/FormationWebservice.zip
Lien 160 : http://acesyde.developpez.com/tutoriels/android/web-services-android/
Lien 161 : http://cyril-mottier.developpez.com/tutoriels/android/introduction-styles-themes/
Lien 162 : http://developer.android.com/reference/android/R.styleable.html
Lien 163 : http://developer.android.com/reference/android/R.style.html
Lien 164 : http://developer.android.com/reference/android/R.attr.html
Lien 165 : http://nicroman.developpez.com/tutoriels/android/styles_themes/