Partie 3

Séparer le contenant et le contenu : L'utilisation de la classe XML

Le nouveau type natif XML dans actionscript 3 recoupe exactement la norme en usage pour un xml externe.
Il est donc possible de charger un XML et l'utiliser comme fichier de contenu et ou de paramétrage en conservant son aspect natif.

Il est possible aussi d'en créer un de toute pièce et de le transmettre à un script php qui se chargera de le transformer en un fichier sur le serveur exploitable par la suite.

L'avantage d'un xml pour une application flash est aussi de libérer les ressuources du serveur et de la base où son utilisation est fortement sollicitée (chat, jeu en reseau…). Pour une galerie d'images par exemple quel besoin de faire une requête sur le serveur si la base n'est pas mise à jour par des contributions d'utilisateurs ? Toutes les opérations se passeront du coté client, optimisant ainsi la bande passante de manière considérable.

Enfin le parcours d'un arbre xml à été entièrement revu en actionscript 3, et est particulièrement pratique comparé à l'ancienne version ou même à javascript.

Structure d'un XML

```
<?xml version="1.0" encoding="utf-8" ?>
<nœudprincipal>
	<enfant propriete= « chaine »>
		<sous_enfants><a class= »monstyle »>cliquez pour voir !</a></sous_enfants>
		<sous_enfants></sous_enfants>
	< /enfant>
</nœudprincipal>
```

Un xml est une liste ordonnée par des balises imbriquées de la même façon que des balise html. A sa différence il n'existe pas de balises prédéfinies, leurs noms et l'arborescence (pas de chevauchement) sont laissés au choix du programmateur (pas de caractères spéciaux ,ni accents, ni espaces). Il existe toutefois des syntaxe xml standardisées permettant l'échange d'informations normées (gestion informatisée des bibliothèques, flex, flux rss…). Un xml est principalement utilisé pour le transport de contenu écrit, et référentiel.

Les éléments

Un xml se compose nécessairement d'un nœud racine dans lequel est déployée le reste de l'arborescence.
Le nœud racine contient le reste de l'arborescence. Il ne peut pas contenir de texte non entouré de balises.
Un noeud ou élément sans enfant peut être noté selon la syntaxe de balise abrégée :
<balise identifiant= « .3 » />

En syntaxe AS3 on parcourt les éléments selon la syntaxe pointée et l'utilisation des crochets pour parcourir des enfants multiples.
Par exemple :
Nœudprincipal.enfant.sous_enfants[0] : cible le contenu du premier nœud sous_enfants.

Les éléments peuvent contenir des chaines de caractères au format HMTL. Il est donc possible de placer dans ces balises du contenu texte formaté à l'aide des balises et des styles css

Les attributs

Tout comme la syntaxe xhtml , il est possible de définir des attributs à l'intérieur des balises ouvrantes des éléments. Les attributs sont une paire d'identifiants et de valeurs.
L'identifiant : doit être nommé selon une syntaxe sans caractères spéciaux (sauf l'underscore), ni espaces, ni accents.
La valeur doit se trouver entre guillemets ou apostrophes. Elle ne peut pas contenir de syntaxe à balise (donc pas de texte formaté en html) mais l'utilisation des slashes est autorisée. La valeur est transmise à flash sous forme de chaine, il est donc nécessaire de faire attention lors par exemple de l'envoi d'un booleen pour que les valeurs soient justes.
On récupère un attribut dans flash à l'aide du signe @ (arobase).
Dans notre exemple du début il suffit de cibler :
Nœudprincipal.enfant.@propriete

Note Importante : un fichier XML destiné à être utilisé par flash doit être impérativement codé en UTF-8
On trouvera l'option adéquate dans l'éditeur de script (via flashDevelop : *file / encoding / UTF-8)* L'entête doit préciser l'encodage utilisé.

Ecrire un xml dans l'espace actionscript

XML est un type primitif. A ce titre il s'instancie de la manière suivante :

```
var parametres : XML =
<params>
        <couleurs fond="0xCCFF33" bords="0xFFFFFFF"></couleurs>
        <alphas fond=".9" bords=".7 »></alpha>
        <titre><fr><u>Notre nouvelle adresse</u></fr><en><u>Our new
adress</u></en></titre>
<params>
```

Hormis sa mise en variable, et l'amputation de l'entête, on ne distingue aucune différence entre l'écriture d'un XML externe ou interne.

Parcourir un fichier XML dans actionscript 3

Le parcours d'un fichier XML à été entièrement revu en AS3 par rapport à la précédente version.

Un fichier XML partage certaines caractéristiques communes avec un tableau qui rendent son parcours un peu similaire.
Le nœud principal dans le XML est remplacé par le nom de la variable :

```
var donnees :XML=
<main>
<parametres ><couleurs fond="0xCCFF22"></couleurs></parametres >
<medias>
<image chemin="images/full/01.jpg "><titre type="html" >Nouveau
batteur</titre></image >
<image chemin="images/full/02.jpg " ><titre type="html" >Racleur à
tomate</titre></image >
</medias>
</main> ;

//Cibler un nœud (ici la balise paramètres) (nous laissons ici le type
indéterminé):
var parametres : * = donnees.parametres;
trace ("parametres : "+parametres) // renvoie la balise complète

//Cibler un identifiant de balise (ici la couleur du fond) :
var couleur_fond :Number = donnees.parametres.couleurs.@fond ;
trace("couleur : "+couleur_fond) // renvoie l'équivalent décimal de la
valeur hexa rentrée.
//. Note sur le transtypage : En imposant une chaine de caractères (String)
// à une variable de type Number, l'application va automatiquement
interpréter la chaine en tant que   nombre .
// Cela ne génère donc pas d'erreurs.
//Cibler une balise dans un nœud multiple(nous récupérons le l'url de la
première référence d'image du xml) :
var ref_image_1 :String=donnees.medias.image[0].@chemin ;
trace("chemin première image : "+ref_image_1)// renvoie :
images/full/01.jpg
//cette notation nous permet d'utiliser une boucle sur un nœud multiple
pour parcourir celui-ci.
//Si nous voulons effectuer un parsage préalable de tous les enfants d'un
nœud nous pourrons soit utiliser une boucle " for " soit comme  ci-dessous
une boucle " for each "

for each (var image :XML in donnees.medias.*)
{
      //Nous verifions si ce noeud possede encore des sous-enfants (noter
l'utilisation de parenthèse après "length))
      If ( image.length()>0 )
      {
            for each(var infos :XML in image)
            {
//Si c'est le cas nous verifions quelles infos sont présentes en nous servant du
despcripteur de balise
                  var chaine_info : String = infos.*.name();

//Notons que le paramètre "name" est ici une methode et prend donc des
//parenthèse à la fin de l'expression ne pas confondre avec la propriété
//name d'un membre du package display.
                  if(chaine_info == "titre")
                  {
                        trace("titre ="+infos.*)
                  }
            }
      }
}
```

Charger un fichier XML externe

Nous utilisons pour cela la classe prédéfinie de chargement de fichiers texte URLLoader() (Nous nous en servons également pour un fichier de type css ou texte simple).

```
//XML dans lequel va être stocké le XML chargé
var dataXML : XML;

//Loader qui va charger le XML
var loader:URLLoader;

var diffuseur : EventDispatcher = new EventDispatcher() ;

/*-- Fonctions de chargement des données --*/

//Fonction de chargement du XML de base
function load(url:String):void
{
	//création du loader et chargement des données
	loader = new URLLoader(new URLRequest(url));
	// ajout d'un ecouteur d'évenement déclenché à la fin du
chargement loader.addEventListener(Event.COMPLETE, finChargement);

}

//Fonction déclenchée à la fin du chargement du XML
function finChargement(evt:Event):void
 {
	//Stocke le XML chargé dans le XML prévu à cet effet
	dataXML = new XML(evt.target.data);
}
```

Note : le code ci-dessus fournit l'architecture minimum pour gérer un téléchargement. Une mauvaise adresse peut générer une erreur qui fera planter l'application. Il est préférable de prévoir une gestion des erreurs.