Guide de référence des fichiers de configuration XML

Présentation

Ce guide présente étape par étape la manière de créer un fichier de configuration XML pour le serveur des entrées/sorties.

Etape 1: *Création d'un fichier de configuration «vide»*

iodef.xml

```
1. <?xml version="1.0"?>
2. <simulation/>
```

XML 1 : Configuration vide de base.

Rappel: Notions du langage XML

* L'abréviation XML signifie e**X**tensible **M**arkup **L**anguage.
* C'est un langage informatique de balisage générique qui sert, dans notre cas, à stocker des données textuelles Unicode.
* Il existe plusieurs versions du langage XML, celle que nous utilisons est la version 1.0 publiée en février 1998 (c'est la plus répandue).
* Le XML est basé sur le modèle d'un arbre, qui comprend plusieurs nœuds de différents types: document, éléments, textes, ... (le rôle de chacun d'entre eux sera expliqué au fur et à mesure).
* Le **nœud de type document** est l'élément racine de l'arbre, il est unique et peut éventuellement avoir des enfants.

Pour créer un document XML valide, on commence par écrire l'entête du fichier (appelé *prologue* dans la nome) qui indique la version du langage que nous allons utiliser. Cette information est obligatoire et sera dans notre cas toujours la même, c'est à dire:
`<?xml version="1.0"?>`
Une autre information facultative peut être ajoutée, il s'agit de l'encodage textuel utilisé dans le fichier. Là encore, nous utiliserons toujours la norme *UTF-8* qui permet de gérer tous les caractères unicodes:
`<?xml version="1.0" encoding="UTF-8"?>`

On indique ensuite le nœud document qui est la racine unique de l'arborescence XML. Ce nœud a toujours pour nom la chaîne «simulation» dans le fichier de configuration principal, quelque soit le cas de figure dans lequel on se place.

Ce nœud peut ne pas avoir d'enfant (auquel cas aucune sortie n'est paramétrée statiquement), son nom est insensible à la casse et ses attributs éventuels ne sont pas analysés par le «parseur XML».
`<simulation/>` ← *valide.*
`<SimuLatIon/>` ← *valide, car insensible à la casse.*
`<simulation id="ma_simulation" />` ← *valide mais l'attribut nommé «id» ayant pour valeur "ma_simulation" ne sert à rien ...*
`<simmulation/>` ← *invalide, le nœud document doit avoir pour nom «simulation». Un avertissement est transmis à l'utilisateur mais l'exécution se poursuit.*

Dans le cas ou le noeud document dispose de nœuds enfants, il s'écrit sous cette forme.
`<simulation>....</simulation>` ← *valide.*
`<SimuLatIon>....</SimuLatIon>` ← *valide, car insensible à la casse.*
`<SimuLatIon>....</simulation>` ← *invalide, car les noms de la balise ouvrante et de la balise fermante sont incompatibles au niveau du traitement XML. Une erreur du type «Tag mismatch» est générée au moment de l'exécution signalant également la position du problème. Le traitement s'interrompt inopinément.*

Notes :

1. Dans la suite de ce document, on admettra qu'un avertissement n'interrompt pas l'exécution du programme contrairement à la détection d'une erreur, laquelle est susceptible de modifier trop profondément le comportement du code par rapport aux souhaits par l'utilisateur.
2. On n'admettra aussi que les noms des nœuds sont insensibles à la casse, contrairement aux valeurs qui leur sont associées. Ex: `<context id="id1" />` ↔ `<CONTEXT ID="id1" />` ← *valides tous les deux.*
3. La présence de deux nœuds racines génère une erreur de type «Junk after document element».
4. Pour le moment, l'absence de prologue est tolérée bien que fortement déconseillée pour respecter les standards.
5. Attention à la position des barres obliques au niveau des balises, une erreur du type *«Tag mismatch»* est générée en cas de problème à ce niveau.
6. La présence de guillemets simples à la place de guillemets doubles est acceptée. Les caractères erronés rencontrés dans le fichier génère une erreur «Invalid token».

Etape 2: *Création des contextes de sorties.*

iodef.xml

```
1.  <?xml version="1.0"?>
2.  <simulation>
3.    <!-- Ici, je commente l'ajout des éléments -->
4.    <context id="c1"/><context id="c2"/>
5.  </simulation>
```

XML 2 : Configuration valide comprenant quelques contexts.

Rappel (suite): Notions du langage XML

* Les retours à la ligne et la présence d'une indentation particulière n'ont pas de signification dans le langage XML mais permettent une lecture plus aisée.
* Le **nœud de type commentaire** est délimité par <!-- et --> et n'est pas interprété par le programme.
* Le **nœud de type attribut** est toujours associé à un autre nœud.
  Il est composé d'un nom et d'une valeur présentés sous la forme *nom="valeur"*.
  Pour un nom d'attribut et un noeud donnés, un attribut est toujours unique.
* Le **nœud de type élément** est probablement le plus utilisé dans un document XML.
  Il dispose d'un nom qui le qualifie et accepte comme enfants tous les autres types de nœud.

Le nœud racine peut disposer d'un nombre quelconque d'éléments enfants nommés «context».
Les contextes permettent de regrouper globalement des conditions de génération de fichier en fonction de situations rencontrées par l'utilisateur, comme la différenciation des sorties lors du passage d'un code de calcul à un autre (pour modèle atmosphérique, modèle océanique, … par exemple).
Chaque contexte doit être identifié par ajout d'un attribut «id» pour être traités.

Quelques cas de figures:

<context id="mon_id" /> ← *valide.*
<conttext id="mon_id">...</conttext> ← *incorrect mais valide. Si le nom de l'élément n'est pas «context», son contenu n'est toutefois pas ignoré car le nœud racine «simulation» ne peut contenir que des nœuds «context». Un avertissement est transmis pour signaler le problème syntaxique.*
<context id="mon_id" name="mon_nom" /> ← *valide même si l'attribut «name» n'est pas nécessaire, un avertissement est transmis à l'utilisateur.*
<context id="mon_id" id="mon_autre_id" /> ← *invalide, à cause de la non-unicité de l'attribut «id» sur le nœud, une erreur de type «Duplicate attribute» est affichée par le programme.*
<context>...</context> ← *valide, mais génère un avertissement, l'absence de l'identifiant empêche la récupération du contexte pour le traitement des sorties qui lui sont associées.*

A propos des identifiants … et de leurs valeurs

Un identifiant doit être composé de caractères alphanumériques ([A-Z][a-z][0-9]) et de tirets bas (_) sans quoi l'identification échoue et un avertissement est renvoyé.
Il est conseiller d'identifier systématiquement les éléments de groupe ainsi que les éléments finaux sauf dans le cas des références à des champs lors de la définition de fichiers (voir la suite). L'absence d'identification entraîne l'impossibilité d'accéder dynamiquement à certains objets du programme par leurs identifiants et peut donc provoquer un comportement inattendu au niveau du traitement des sorties.

Etape 2 bis: *Ajout des groupes de définitions associés aux contexts.*

iodef.xml

```
1.  <?xml version="1.0"?>
2.  <simulation>
3.    <!-- Définitions des différents contextes d'E/S -->
4.    <context id="c1">
5.      <!-- Ajout des groupes de définitions -->
6.      <field_definition/>   <file_definition/>   <axis_definition/>   <grid_definition/>
7.    </context>
8.    <context id="c2"/>
9.  </simulation>
```

XML 3: Configuration valide avec groupes de définitions.

Etape 3: *Définitions des grilles et des groupes de grilles.*

iodef.xml

```
1.  <?xml version="1.0"?>
2.  <simulation>
3.    <!-- Définitions des différents contextes d'E/S -->
4.    <context id="c1">
5.      <!-- Liste des groupes de définitions -->
6.      <field_definition/>
7.      <file_definition/>
8.      <axis_definition/>
9.      <grid_definition>
10.       <!-- Définitions des grilles et groupes de grilles -->
11.       <grid_group id="ggr1">
12.         <grid id="gr1" description="desc. gr1" />
13.         <grid id="gr2" name="nom gr2"/>
14.       </grid_group>
15.       <grid id="gr3" name="nom gr3" description="desc. gr3"/>
16.       <grid id="gr4" />
17.     </grid_definition>
18.   </context>
19.   <context id="c2"/>
20. </simulation>
```

XML 4: Configuration valide incluant la définition des grilles.

Liste des attributs de l'élément grid_group

Nom de l'attribut	Type de donnée	Héritage	Valeur par défaut	Description de l'attribut
id	**string**	/	/	Identifiant de l'élément dans l'arborescence
name	**string**	Ref	/	Nom de l'élément (facultative).
description	**string**	Ref	/	Description de l'élément (facultative).
??? pour le reste (zoom, domain, rank, etc...) ???				

Etape 4: *Définitions des axes et des groupes d'axes.*