Programmation des arbres de décision

mise à jour : 05/06/2016

Partie 1 : Présentation générale

1 - Introduction

Un arbre de décision (AD) est un outil d'aide à la décision qui représente un ensemble de choix sous la forme d'un arbre graphique.
Les différents résultats possibles sont atteints en fonction des décisions prises à chaque étape.

Les AD contiennent des branches qui constituent le cheminement opérationnel : meilleure stratégie dans un contexte donné.
Ce cheminement est implicitement formalisé par un expert du domaine.

En pratique, une succession de questions est posée et les réponses apportées réalisent un contexte.
Les AD sont de véritables moteurs logiques du type : si contexte C1 alors décision D1.

Un arbre de décision informatisé (ADI) va reproduire l'arbre graphique et proposera en outre les questions nécessaires à l'aide de formulaires.

L'intérêt d'un arbre décisionnel informatisé réside dans sa simplicité d'utilisation, compatible avec un exercice professionnel en temps réel, lors d'une consultation.
Le système lit le fichier contenant l'arbre décisionnel puis exécute cet arbre. Il pose alors des questions et signale ses conclusions.
L'ensemble des informations produites (le chemin) est rassemblé dans une fenêtre du système sous forme textuelle ou graphique, ce qui permet d'avoir un regard critique sur l'ensemble du processus.

Les AD peuvent être utilisés de 2 façons :

1- pour la conduite à tenir face à un problème connu mais complexe à résoudre car dépendant de multiples paramètres.
Il s'agit alors d'une application autonome.

Ex : que faire devant un taux de cholestérol donné, une HTA avec facteurs de risque, une suspicion d'ostéoporose ...

2- pour l'aide au diagnostic (en association avec un système expert).

Ex : aide au diagnostic d'une anémie

L'entrée de données dans le système expert (SE) permet de conclure à une anémie.
Il faut alors entrer les bonnes données pour arriver à un diagnostic. Pour éviter une perte de temps le SE exécute un arbre de décision dont les questions sont celles d'un expert du domaine pour arriver rapidement à un diagnostic.
L'arbre pose donc des questions pour trouver la cause de l'anémie. Les réponses sont injectées dans le SE ce qui permet de générer des conclusions voir des questions nouvelles, ou d'autres voies pour le diagnostic final.
Il y a échanges entre SE et AD.

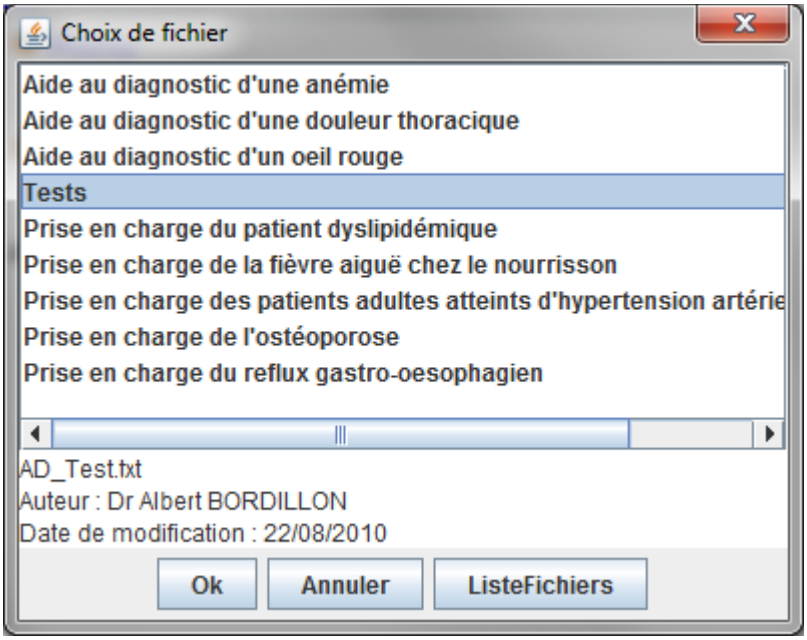
2 – Exécution d'un AD

Un AD est contenu dans un fichier au format texte (extension 'txt') qu'il suffit de charger puis d'exécuter.

Bouton de chargement de fichier.

Le choix du fichier à charger est fait dans une boite qui affiche les titres des fichiers AD disponibles.

Ex : Le fichier 'Tests' a été choisi (clic dans la liste) - Cliquer sur le bouton 'Ok' pour charger le fichier 'AD_Test.txt'



Quand le chargement est terminé (moins d'une seconde) l'arbre graphique est affiché.
L'AD peut être exécuté.

	Cliquer sur ce bouton pour exécuter l'arbre

La première branche de l'arbre est alors exécutée.
L'exécution des branches suivantes est contrôlé par des instructions de saut elles même dépendantes des réponses aux questions posées.

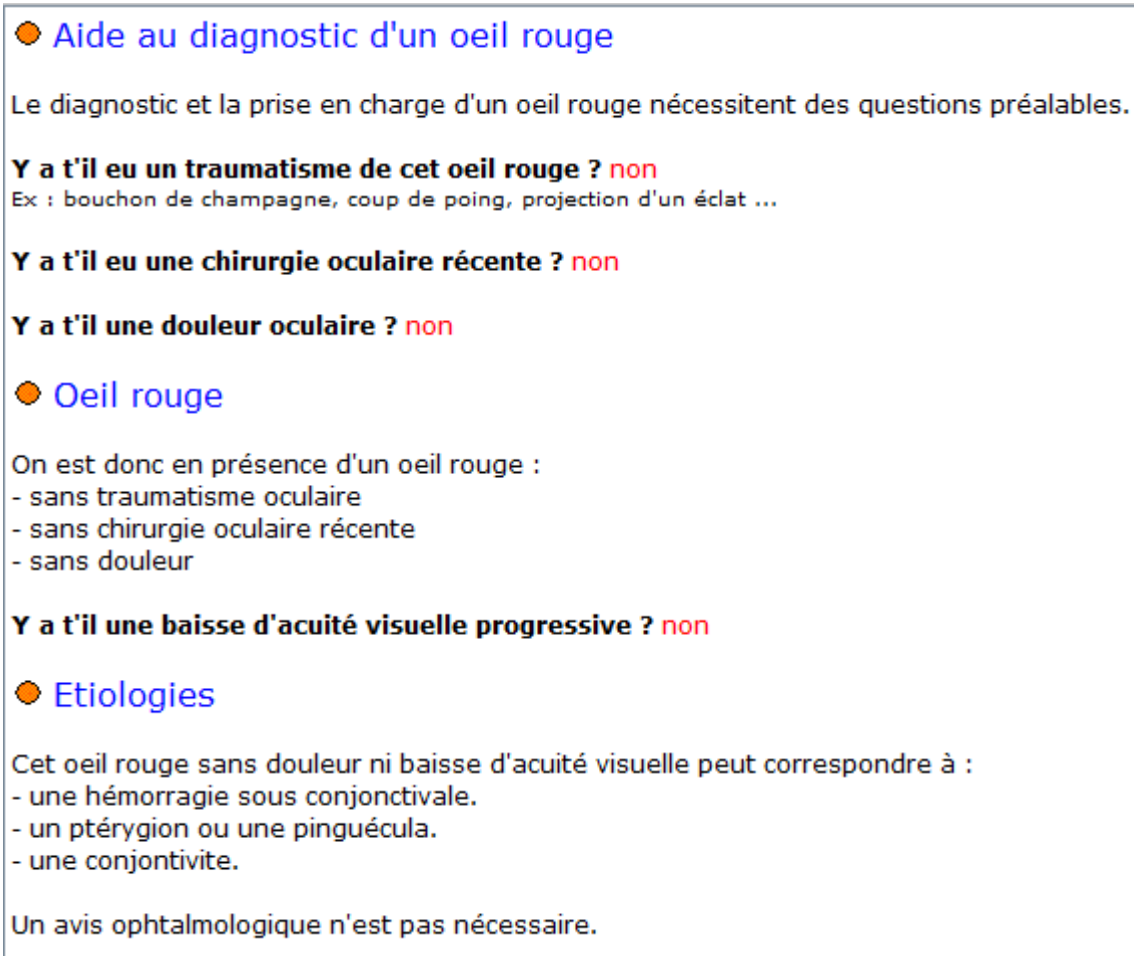
L'exécution d'une branche provoque l'affichage d'une boite de dialogue contenant le premier formulaire.

3 – Le chemin

- **chemin textuel**

Les informations et questions des formulaires sont affichées dans des boites de dialogue mais également dans une zone de texte qui restitue le chemin suivi dans l'arbre.
Il est ainsi possible d'avoir une vision critique de tout le processus suivi.

Ex :

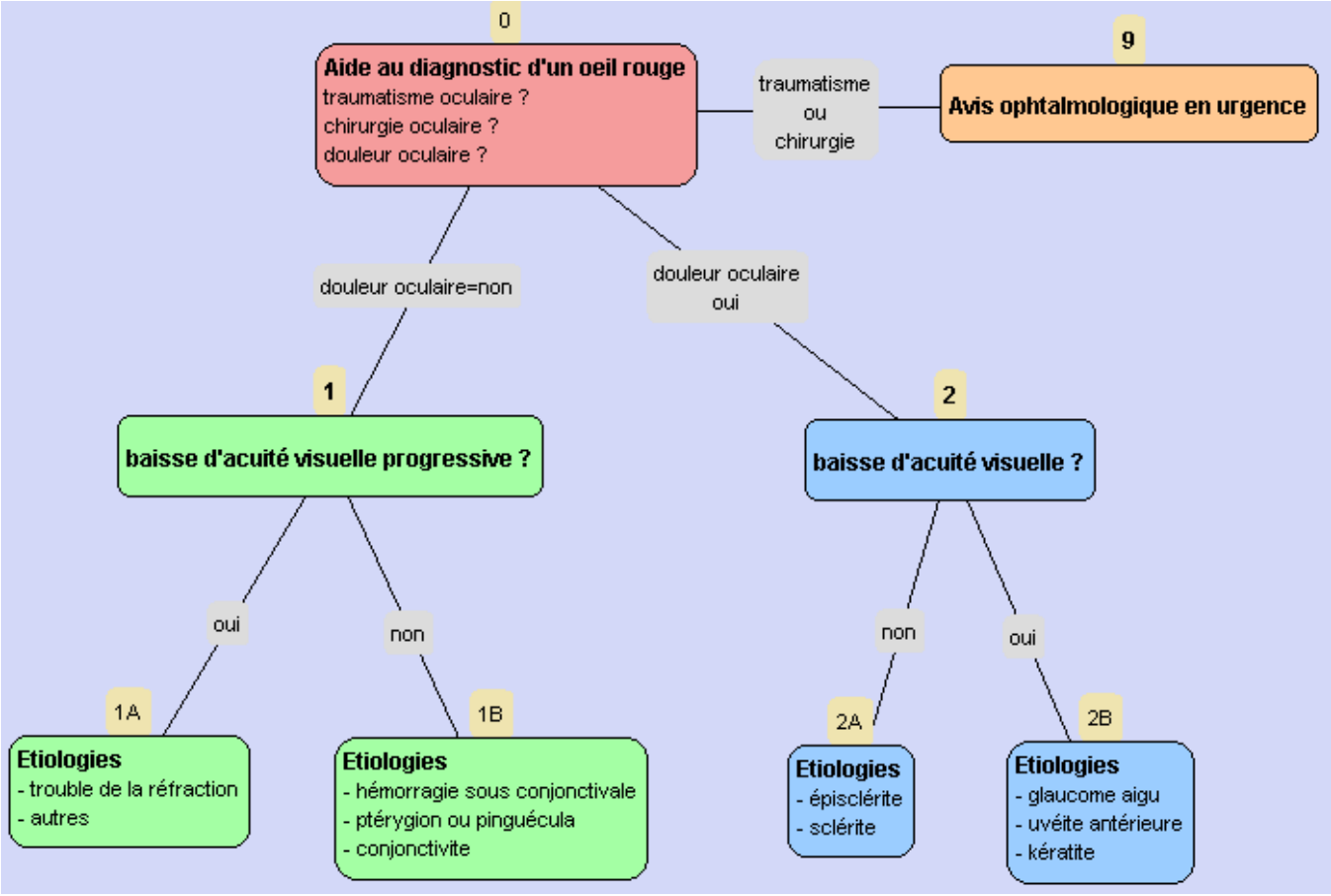


Les titres des branches sont affichés en bleu, précédés d'un point orange.
Les questions sont suivies de leur réponse en rouge.

Ce chemin peut être exporté dans un fichier au format HTML

- **chemin graphique**

Le chemin parcouru dans l'arbre est coloré en jaune.
On voit bien le trajet emprunté selon les réponses aux questions précédentes.



Partie 2 : Programmation

Nous allons voir successivement :

- Des principes généraux
- La balise <head>
- La balise <branche>
- La balise <aide>
- La balise <formulaire>
- Les instructions des formulaires
- Les balises des formulaires
- Les actions des instructions des formulaires
- La balise <scriptFormulaire>
- La balise <scriptBranche>
- La balise <include>
- La balise <importation>
- Les variables

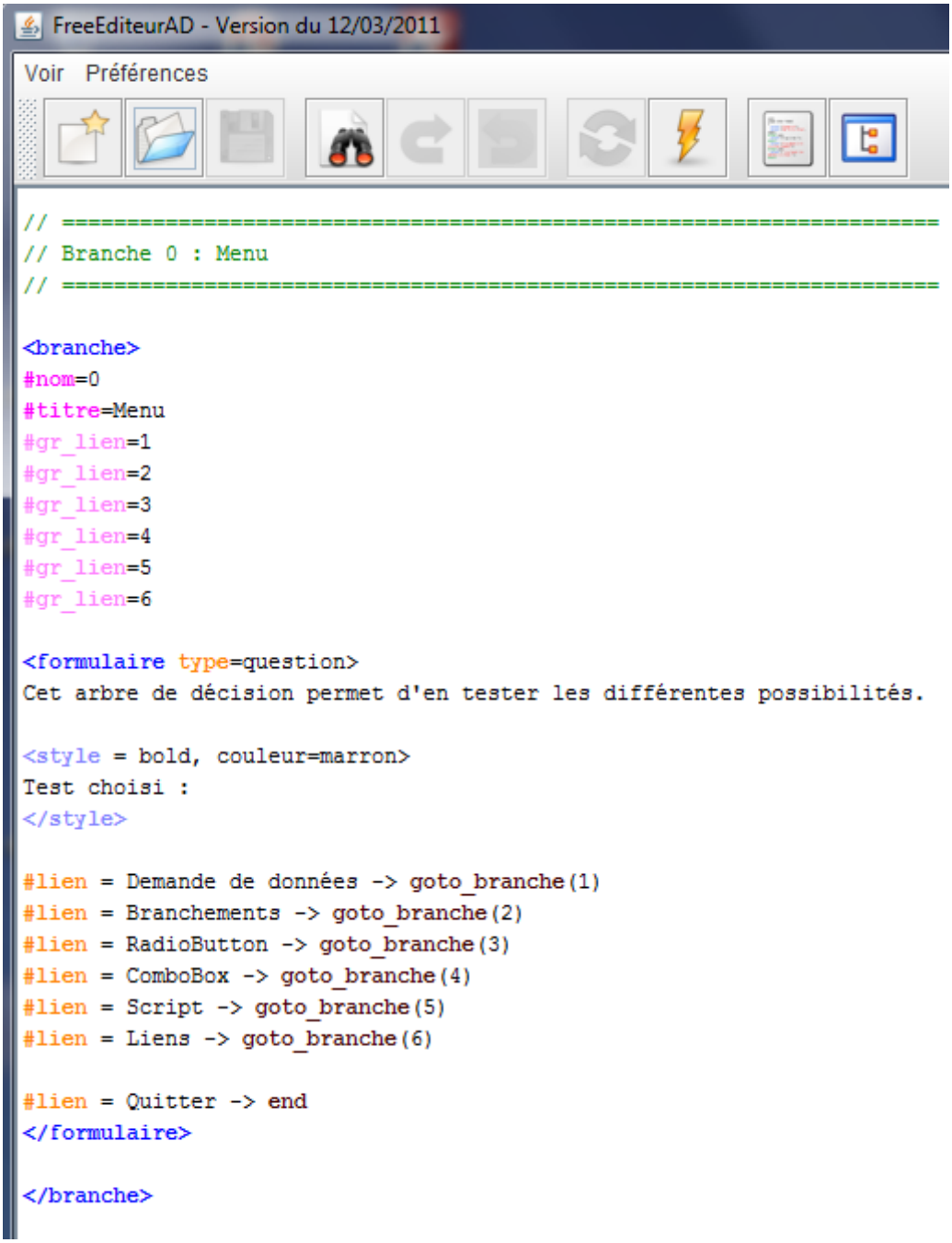
1 – Principes généraux

Dans cette partie nous allons apprendre à créer des AD ce qui revient à rédiger des fichiers AD.
Un fichier AD est un fichier au format texte (extension 'txt') qui peut être créé avec un simple éditeur de texte : bloc notes Windows ou WordPad.

Il est cependant souhaitable d'utiliser l'éditeur spécialisé FreeEditeurAD qui offre les services suivants :

- un éditeur de texte avec coloration syntaxique.
- une fenêtre visualisant l'arbre sous forme graphique.
- L'exécution du fichier texte qui est testé immédiatement.
- Des vérifications : instructions dans les balises, ...

Voici un exemple de copie d'écran.




```
// ====================================================================
// Branche 0 : Menu
// ====================================================================

<branche>
#nom=0
#titre=Menu
#gr_lien=1
#gr_lien=2
#gr_lien=3
#gr_lien=4
#gr_lien=5
#gr_lien=6

<formulaire type=question>
Cet arbre de décision permet d'en tester les différentes possibilités.

<style = bold, couleur=marron>
Test choisi :
</style>

#lien = Demande de données -> goto_branche(1)
#lien = Branchements -> goto_branche(2)
#lien = RadioButton -> goto_branche(3)
#lien = ComboBox -> goto_branche(4)
#lien = Script -> goto_branche(5)
#lien = Liens -> goto_branche(6)

#lien = Quitter -> end
</formulaire>

</branche>
```

Le texte contenu dans le fichier est un code qui est exécuté ligne après ligne par l'application pour fournir des actions.
Un tel code est un script qui constitue un petit langage de programmation.

Un arbre est formé de branches contenant des formulaires.

Le fichier AD utilise un système de balises qui délimitent des zones de fichier.
Format d'une balise de début de zone : `<nomBalise>`
Format d'une balise de fin de zone : `</nomBalise>`

Voici un exemple de squelette de fichier AD

```
// ----------------------------------------------------------------------
// AD_Test.txt
//
// Arbre décisionnel pour des tests
// ----------------------------------------------------------------------

<head>
</head>

<arbre>
</arbre>

<branche>
  <formulaire type=question>
  </formulaire>

  <formulaire type=information>
  </formulaire>
</branche>

<branche>
  <formulaire type=commentaire>
  </formulaire>
</branche>
```

Les lignes débutant par // sont des commentaires servant à documenter le code.
Ces commentaires peuvent figurer à tout endroit du fichier, à condition de débuter une ligne.

Ex :

permis	non autorisé
// commentaire	<branche> // commentaire
<branche>	

Il y a une balise <head> par fichier
Il y a une ou plusieurs balises <branche> par fichier
Il y a une ou plusieurs balises <formulaire> dans une balise <branche>

Remarques sur les balises :
Le couple <nomBalise> </nomBalise> est obligatoire.
Les couples de balises doivent être disjoints.

Ex :

permis	non autorisé
<branche>	<branche>
</branche>	<branche>
<branche>	</branche>
</branche>	</branche>

2 – La balise <arbre>

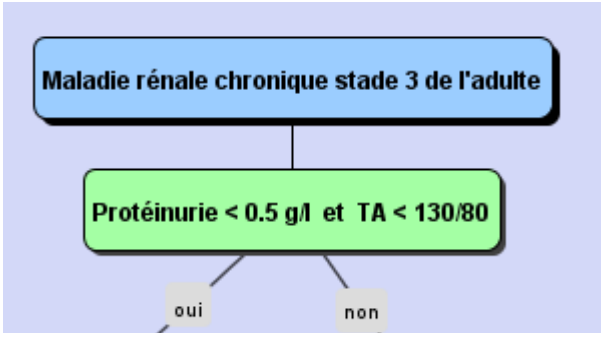
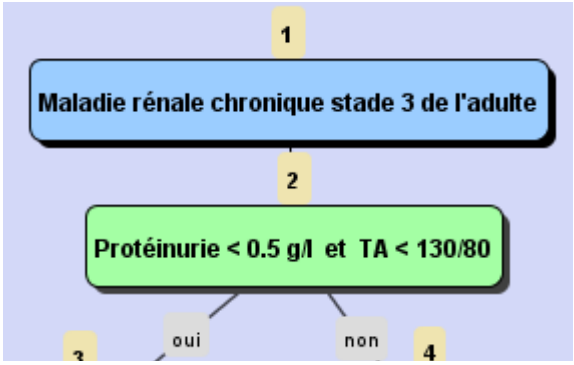
Contient des instructions capables de modifier le fonctionnement global, par défaut, des arbres de décision.

Ex :

```
<arbre>
#dessin_étiquettes=non
#présentation=diagramme
#exécutable=non
</arbre>
```

- **instruction #dessin_étiquettes = oui/non**

Permet de dessiner ou non les étiquettes jaunes contenant le nom des branches.
Par défaut les étiquettes ne sont pas dessinées.

#dessin_étiquettes=non	#dessin_étiquettes=oui
	

- **instruction #présentation = chemin/diagramme**

Permet de choisir la représentation initiale des arbres de décision sous forme d'un chemin textuel ou d'un diagramme.
Par défaut les AD sont présentés sous forme de diagramme, comme dans l'exemple plus haut.
La présentation sous forme d'un chemin textuel permet des voir les questions posées et les réponses retournées.
L'utilisateur peut choisir une représentation à l'aide des boutons de l'application.

- **instruction #exécutable = oui/non**

Précise si un arbre peut être exécuté ou non.

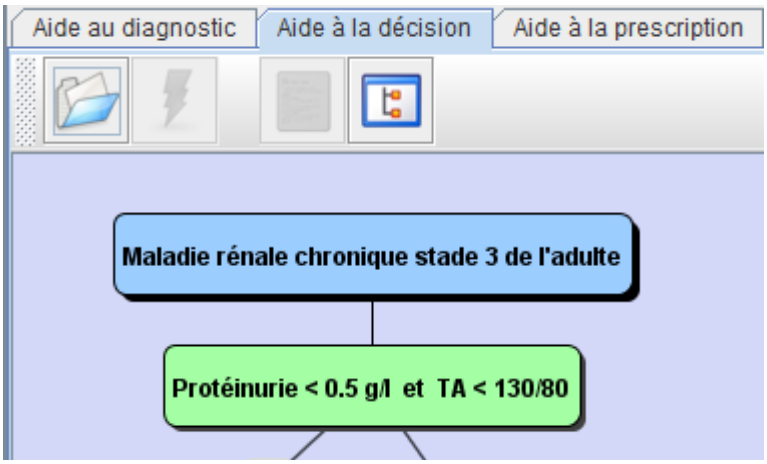
Par défaut, un arbre est fait pour être exécuté.
Les différents formulaires sont alors exécutés, affichant des informations et posant des questions.

Dans le cas d'un arbre purement graphique, ne posant pas de questions, l'exécution de cet arbre n'est pas nécessaire. Il suffit de le voir. Si on exécute cet arbre rien ne se passe.

L'instruction ''exécutable=non'' est alors intéressante car elle permet de désactiver le bouton d'exécution de l'arbre. L'utilisateur n'est pas dérouté par l'absence d'action lors d'une tentative d'exécution.

Ex :
L'arbre ''Prise en charge d'une maladie rénale chronique stade 3 de l'adulte'' est purement graphique.

```
<arbre>
#dessin_étiquettes=non
#présentation=diagramme
#exécutable=non
</arbre>
```



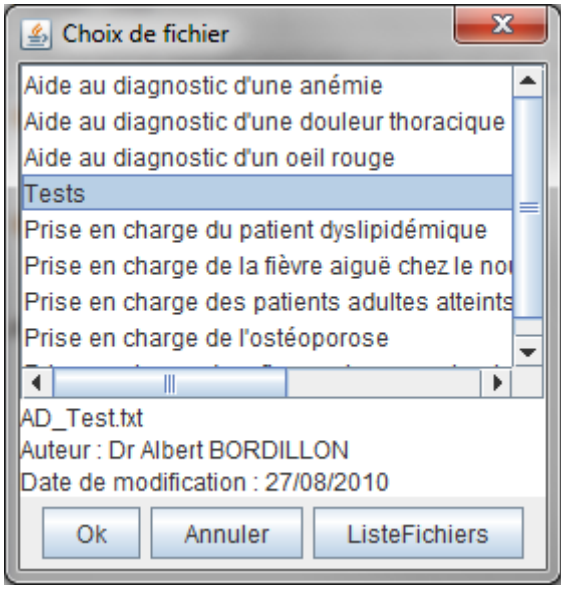
Le bouton d'exécution de l'arbre, ainsi que celui de visualisation du chemin textuel ont été désactivés.

3 – La balise <head>

Cette balise permet de renseigner la boite de choix de fichier et le système.

Ex :

```
<head>
#Date de création      = 25/02/2010
#Date de modification = 22/08/2010
#Auteur = Dr Albert BORDILLON
#Titre = Tests
#Validation = oui
#Version = 1.0
#Système d'unités = ancien
</head>
```



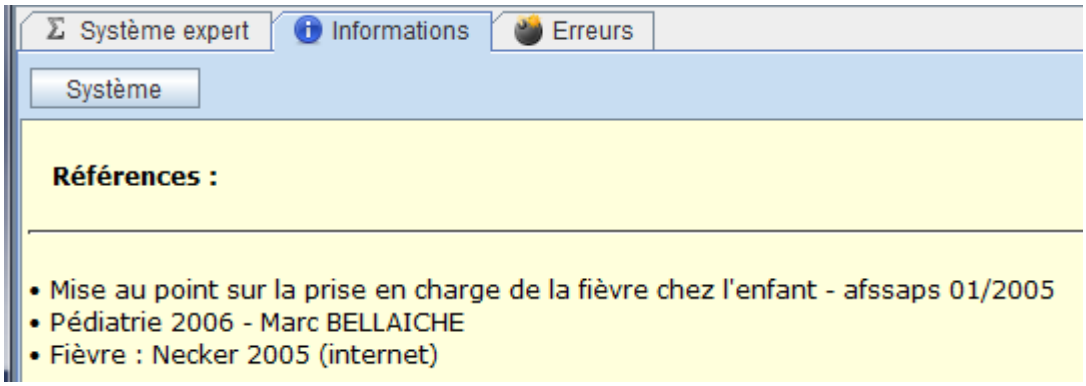
La boite affiche les titres de tous les fichiers AD disponibles.
Un clic sur une ligne de titre (Test) affiche en bas de la boite des informations sur le fichier.

Les instructions ne sont pas toutes indispensables mais pourront servir dans une version ultérieure.

#Date de création	Date de création du fichier
#Date de modification	Date de modification du fichier
#Auteur	Auteur du fichier
#Domaines	Ex : cardiologie, infectiologie (pour une future recherche)
#Mots clefs	Ex : péricardite (pour une future recherche)
#Titre	(Test) apparaît dans la liste des choix de fichiers
#Référence	Ex : nom auteur – nom revue – date (il peut y avoir plusieurs lignes #Référence)
#Validation	oui si l'arbre a été validé par des utilisateurs experts.
#Version	(1,0) pour des mises à jour ultérieures de format.
#TypeFichier	AD = arbre décisionnel.
#Système d'unités	ancien/SI – système d'unités utilisé par l'auteur du fichier. (1)

Les références sont affichées lors du chargement de l'AD dans l'onglet Iformations de freeSAM

#Référence = Mise au point sur la prise en charge de la fièvre chez l'enfant - afssaps 01/2005
#Référence = Pédiatrie 2006 - Marc BELLAICHE
#Référence = Fièvre : Necker 2005 (internet)



4 – La balise <branche>

Ex :

```
<branche>
#nom=5
#titre=Test Script

<formulaire type=question>
</formulaire>

<formulaire type=information>
</formulaire>

<scriptBranche>
</scriptBranche>

</branche>
```

Une branche peut contenir :

- des instructions.
- une ou plusieurs balises <formulaires>
- une ou zéro balise <scriptBranche> en fin de branche

- **instruction #nom = xxx**

Le nom attribué à chaque branche doit être unique. Il est utilisé en particulier dans les instruction de saut vers les branches. Il est donc conseillé d'utiliser un nom bref (ou un nombre)

- **instruction #titre = xxx**

Le titre est le premier texte affiché dans la boite de dialogue

Ex :

```
<branche>
#nom=5
#titre=Test Script
```

- **instruction #gr_couleur = vert / orange / bleu / violet / rose / gris / blanc**

Couleur du nœud de l'arbre graphique.
Si cette instruction est absente la couleur rose est utilisée par défaut.
La couleur jaune est réservée aux nœuds exécutés.

- **instruction #gr_texte = texte1 | texte2 | ...**

Texte(s) affiché(s) dans les nœuds de l'arbre graphique, en complément du titre.
Le caractère '|' provoque un retour à la ligne : voir l'exemple plus bas.
Sans ce caractère '|' il faudrait écrire :
#gr_texte=texte1
#gr_texte=texte2
#gr_texte=texte3

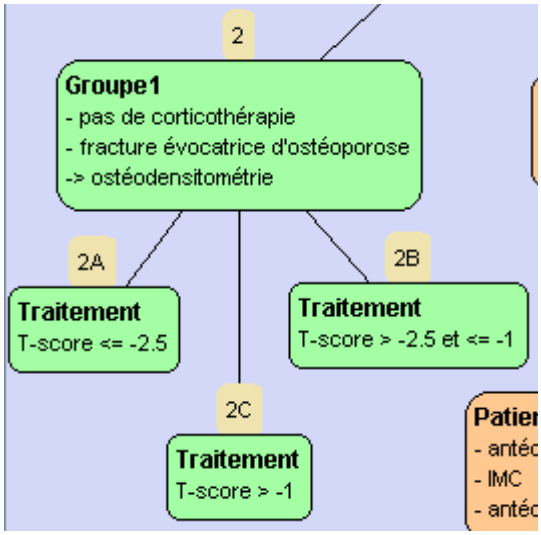
- **instruction #gr_lien = nom d'une branche cible [,texte du lien]**

Ex : #gr_lien=2A
Dessine un lien graphique entre le nœud de la branche en cours et celui de la branche 2A.

Ex : #gr_lien=4F, non
Dessine un lien graphique entre le nœud de la branche en cours et celui de la branche 2A.
Affiche 'non' au milieu du lien.

Voyons ces propriétés dans un exemple :

```
<branche>
#nom=2
#titre=Groupe1
#gr_couleur=vert
#gr_texte=- pas de corticothérapie|-
fracture évocatrice d'ostéoporose
#gr_texte=-> ostéodensitométrie
#gr_lien=2A
#gr_lien=2B
#gr_lien=2C
```



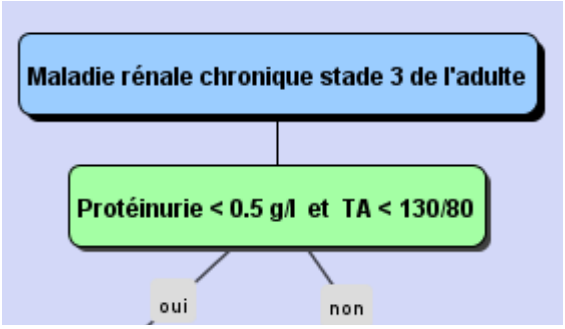
La branche de nom '2' est affichée graphiquement. Le titre 'Groupe1' est en gras
Du texte complémentaire est affiché dans le nœud 2
3 liens partent du nœud 2 (vers 2A, 2B, 2C)

5 – La balise <aide>

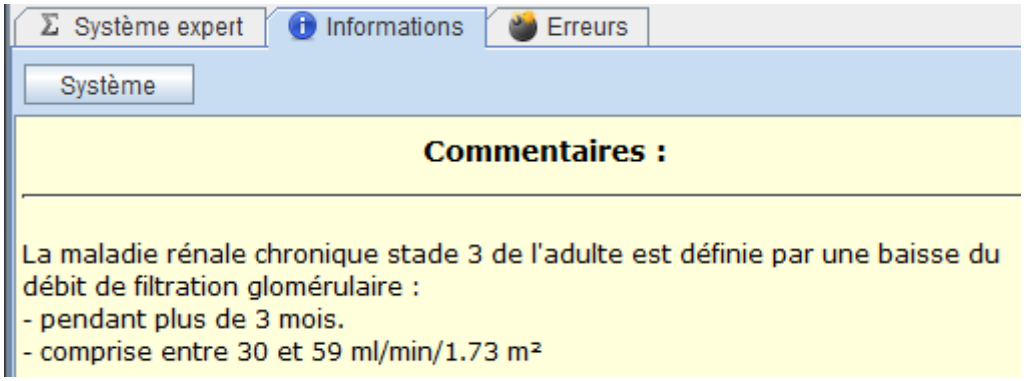
Cette balise permet d'afficher un texte d'aide lors du survol du nœud graphique d'une branche.

Ex :

```
<branche>
#nom=1
#titre=Maladie rénale chronique stade 3 de l'adulte
#gr_couleur=bleu
#gr_lien=2
<aide>
La maladie rénale chronique stade 3 de l'adulte est définie par une baisse du débit
de filtration glomérulaire :
- pendant plus de 3 mois.
- comprise entre 30 et 59 ml/min/1.73 m²
</aide>
</branche>
```



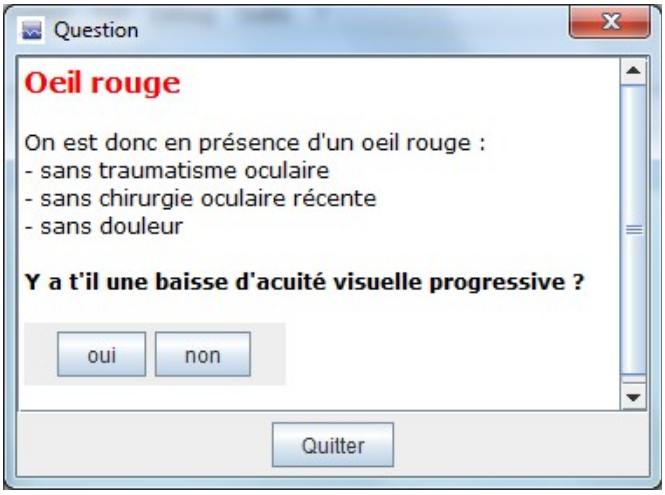
Le nœud graphique (en bleu) comportant une aide est dessiné avec un ombrage noir et épais.
Ceci permet de savoir qu'une aide est disponible.



Le survol (par le curseur de souris) du nœud graphique provoque l'affichage de l'aide.

6 – La balise <formulaire type=xxx>

Un formulaire est une boite de dialogue pour afficher des informations et poser des questions.
Cette boite est créée puis exécutée en suivant ligne après ligne le script du fichier AD.



Un exemple de formulaire en cours d'exécution.

La création du formulaire, et donc de la boite de dialogue, débute avec la balise <formulaire>

Le formulaire est ensuite enrichi par l'insertion de texte et d'éléments actifs : zones de saisie, boutons …
Cette insertion est faite à l'aide d'instructions. La création se termine avec la balise </formulaire>

L'exécution du formulaire débute après la création, avec la même balise </formulaire>

L'exécution rend la boite visible.
Cette boite est modale, c'est à dire que l'exécution de la suite du script est suspendue aussi longtemps que l'utilisateur n'aura pas saisi ses données et cliqué sur un bouton de fermeture de la boite.
La suite du script peut comporter d'autres formulaires. FreeSAM peut sembler bloqué tant que le script n'est pas totalement exécuté.

Le bouton 'Ok' valide les choix faits, ferme la boîte de dialogue et passe à la suite de la branche ou de l'arbre.

Le bouton 'Quitter' termine le cheminement dans l'arbre.
Le titre est affiché en rouge pour les questions, noir pour les informations, gris pour les commentaires.
Les commentaires peuvent ne pas être affichés par choix de l'utilisateur.

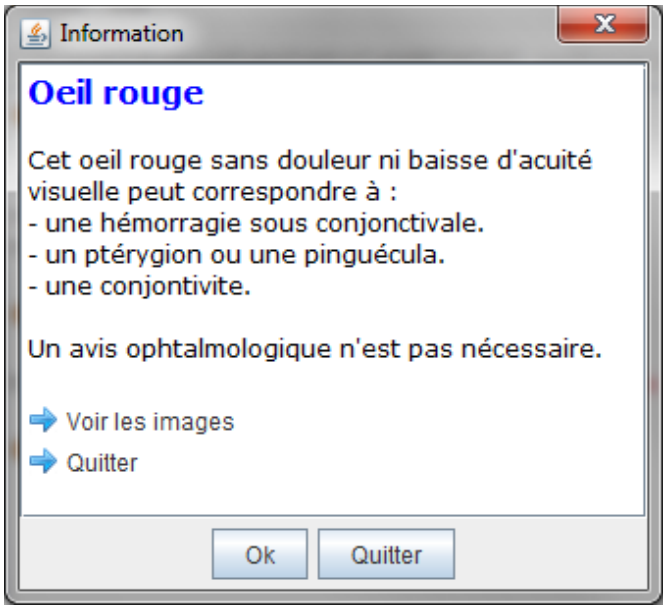
Les formulaires peuvent être de 3 types :

- **<formulaire type=information>**

Ces formulaires sont destinés à l'affichage d'informations (sans poser de question).
Le titre est affiché en bleu.
Il peut y avoir des liens en fin de formulaire pour diriger le cheminement dans l'arbre.

Ex :

```
<branche>
#nom=1B
#titre=Oeil rouge

<formulaire type=information>
Cet oeil rouge sans douleur ni baisse
d'acuité visuelle peut correspondre à :
- une hémorragie sous conjonctivale.
- un ptérygion ou une pinguécula.
- une conjontivite.

Un avis ophtalmologique n'est pas
nécessaire.

#lien = Voir les images -> goto_suite
#lien = Quitter -> end
</formulaire>
```



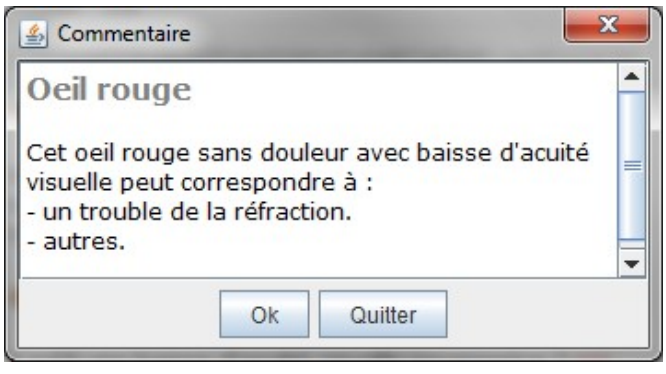
Remarquez les 2 liens en fin de formulaire : Voir les images - Quitter

- **<formulaire type=commentaire>**

Ces formulaires affichent également des informations mais seulement si l'utilisateur le souhaite.
Un menu de l'application permet de ne pas afficher ces formulaires.
Le titre est affiché en gris.

Ex :

```
<branche>
#nom=1A
#titre=Oeil rouge

<formulaire type=commentaire>
Cet oeil rouge sans douleur avec baisse
d'acuité visuelle peut correspondre à :
- un trouble de la réfraction.
- autres.
#actionOK = end
</formulaire>
</branche>
```

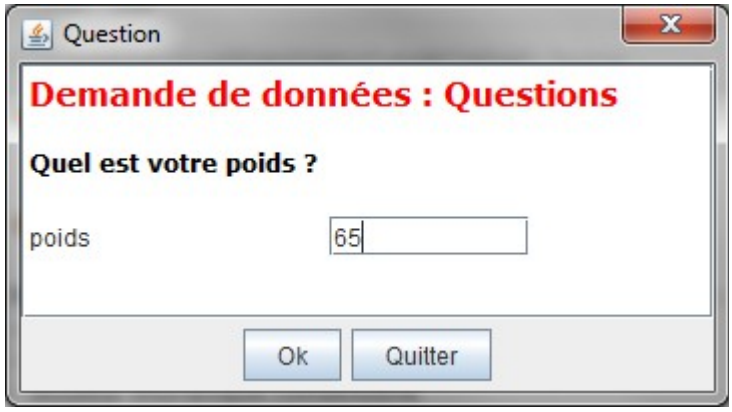


- **<formulaire type=question>**

Ces formulaires permettent de poser des questions à l'utilisateur.
Le titre est affiché en rouge.
Un formulaire peut contenir une ou plusieurs questions.
Une instruction définit le composant visuel collectant les réponses.

Ex :

```
<branche>
#nom=1A
#titre=Demande de données: Questions

<formulaire type=question>
<? Quel est votre poids ?>

#inputTxt = poids -> v_poids
</formulaire>
```



Remarque : le texte de la question est entouré par la balise <?>.
cela permet de le différencier d'un autre texte et de l'afficher en gras.

#inputTxt est une instruction.

Nous allons étudier successivement :
- les instructions des formulaires
- les balises des formulaires
- les actions des formulaires

7 – Les instructions des formulaires

Une instruction est faite pour exécuter une certaine tâche.
Une instruction a le format suivant : #instruction = xxx, yyy, ...

- **#condition = v_X = valeur**

La ou les conditions sont placées en début de formulaire.
Si une des conditions n'est pas respectée le formulaire n'est pas exécuté.

Ex :

```
<formulaire type=question>
#condition = v_sexe = féminin
<? Quelle est la date de vos dernières règles ?>
#inputTxt = date -> v_dateR
</formulaire>
```

Si la variable v_sexe n'est pas égale à 'féminin' le formulaire n'est pas exécuté.
(logiquement la variable v_sexe a été affectée dans un formulaire préalable)

- **#inputTxt = "titre" nom -> v_X toSE4 [action1, actionN]**

Ex :

`#inputTxt = poids -> v_poids`	poids 65

Affiche un libellé (poids) puis une zone de saisie.
La valeur 65 sera affectée à la variable v_poids.

Permet l'affichage d'une zone pour la saisie de données.
La donnée saisie peut être :

- affectée à une variable de l'arbre de décision
- affectée à une variable du système expert SE1
- entrée telle quelle dans le système expert SE4

titre	est un texte affiché avant la zone de saisie
nom	est un texte affiché avant la zone de saisie si titre est absent
nomVariable	est le nom de la variable le l'arbre qui prend la valeur saisie.
toSE4	provoque l'entrée de la valeur saisie dans SE4
[action1, actionN]	actions effectuées lors de la lecture du texte

Si nom est identique au nom d'une variable du système expert (SE) :
- la variable SE est affectée de la valeur saisie
- la zone de saisie est remplie avec la valeur de la variable SE si elle est connue

Actions lors de la lecture du texte (action1, actionN) :

- toMaj	met le texte en majuscules
- delMot(Mot1)	supprime le mot Mot1 du texte.

Exemples de syntaxes possibles :

```
#inputTxt = "quel est le poids ?" poids -> v_poids
```

affiche "quel est le poids ?" puis la zone de saisie
la variable d'arbre v_poids prend la valeur saisie.

'poids' correspond au nom d'une variable contenue dans SGC ->

- l'unité du poids est affichée
- si poids a déjà une valeur -> cette valeur est pré-saisie dans la zone de saisie
- la saisie du poids -> poids est entré comme symptôme dans le SE (Ex : poids=65)

```
#inputTxt = "quel est le poids ?" -> v_poids
```

affiche "quel est le poids ?" puis la zone de saisie.
v_poids prend la valeur saisie.

```
#inputTxt = poids -> v_poids
```

Affiche "poids" puis la zone de saisie. v_poids prendra la valeur saisie.
'poids' est entré comme symptôme dans le SE

```
#inputTxt = "nom de la pilule" -> toSE4 [toMaj, delMot(GE)]
```

affiche "nom de la pilule" puis la zone de saisie
la valeur saisie (Ex : daily Ge) est :

- convertie en majuscule → DAILY GE
- le mot GE est supprimé → DAILY
- entrée dans SE4 → DAILY

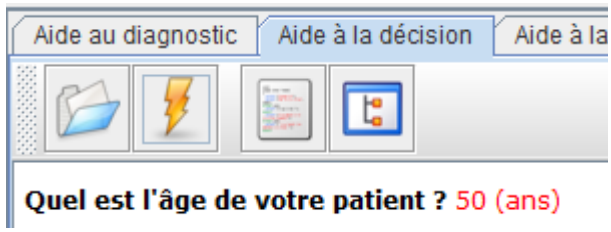
- **#inputBigTxt = nom -> v_X**

Permet de saisir du texte sur plusieurs lignes.
Au début la zone de saisie est composée d'une seule ligne. Il suffit de taper la touche 'entrée' pour créer une ligne supplémentaire.

- **< ? xxx ?>**

Pose une question suivie d'une saisie. Le libellé de la question (xxx) est en gras et précédé d'un point orange.

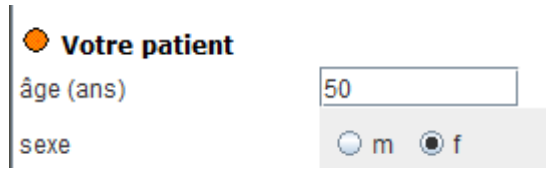
Ex :

<? Quel est l'âge de votre patient ?> #inputTxt = âge -> v_age	
Après saisie de la réponse celle ci apparaît dans le chemin textuel :	

- **#txtTitre = xxx**

Affiche un texte (xxx) comme un titre : texte en gras.
A utiliser à la place de < ? xxx ?> quand plusieurs saisies sont demandées.

Ex :

#txtTitre = Votre patient #inputTxt = âge-> varAge #inputRB = sexe(m,f) -> varSexe	

Les données saisies sont ainsi affichées :	**Votre patient** âge 50 (ans) sexe f

- **#txtAide = xxx**

Affiche une aide pour une saisie avec des caractères de petite taille.

Ex :

`#txtTitre = Identification du patient :`
`#inputTxt = date de naissance -> varDateN`
`#txtAide = format : JJ/MM/AAAA`
	Identification du patient :
	date de naissance
	format : JJ/MM/AAAA

- **#inputRB = nom(valeur1, valeur2, …) -> v_X**

Ex :

`#inputRB=sexe(masculin,féminin)-> v_poids`	sexe masculin féminin

La variable v_poids vaudra 'masculin' si on a cliqué sur le radio bouton masculin.

- **#inputCB = nom(valeur1, valeur2, …) -> v_X**

Ex :

`#inputCB = couleur(bleu,rouge,vert,` `jaune) -> v_couleur`	couleur rouge bleu rouge vert jaune

Un clic sur la ligne 'rouge' de la comboBox -> v_couleur prend la valeur 'rouge'

- **#lien = nom -> action1, action2, ...**

Ex :

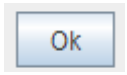
`#lien = Quitter -> end`	Quitter

Le lien est signalé par une flèche bleue.
Un clic de souris sur le mot 'Quitter' provoque l'action 'end' (sortie de l'arbre)
Plusieurs actions peuvent être exécutées, séparées par des virgules.
Les actions des formulaires seront étudiées plus loin.

- **#actionOK = action1, action2, ...**

Ex :

```
#actionOK = goto_branche(2)
```



Le bouton Ok est celui de le boite de dialogue affichant le formulaire (et non pas un bouton supplémentaire)
Normalement, un clic sur ce bouton provoque l'exécution du formulaire suivant (dans la même branche ou la branche suivante)
Ce comportement par défaut peut être changé en provoquant un saut vers une branche donnée.

Action peut être aussi :

- v_X=valeur (attribue une valeur à une variable – Ex : v_age=15)
- end (le cheminement dans l'arbre est interrompu)

- **#item + #groupe**

#item = nom1 -> action1, action2, ...
#item = nom2 -> action1, action2, ...
#groupe = nom, type=xxx, dispo=H/V

un item est un composant visuel d'un groupe défini ensuite.
Il possède : un nom, des actions séparés par des virgules.

Un groupe regroupe les composants visuels (les items)
Il possède des propriétés :
- nom : nom du groupe
- type = bouton -> définit les composant visuels de type bouton
= radio bouton -> affiche des radio boutons
- dispo = H, V : disposition horizontale, verticale des composants

Ex 1 :

```
#item=oui -> v_chaleur=oui, goto_suite
#item=non -> v_chaleur=non, goto_suite
#groupe=temp, type=bouton
```

Le groupe de nom 'temp' contient 2 boutons.
Un clic sur le bouton 'oui' -> la variable 'v_chaleur' prend la valeur 'oui' –> saut vers la suite de l'arbre.
L'action 'goto_suite' permet d'aller vers la suite dès le clic sur le bouton 'oui' sans devoir cliquer ensuite sur le bouton 'Ok' de la boite de dialogue.

Bien entendu, cette action doit être programmée seulement dans la dernière instruction (ou dans le dernier groupe) du formulaire.

Ex 2 :

<? Comment aimez vous les fraises ?> #item = un peu -> v_fraises=1 #item = beaucoup -> v_fraises=2 #item = passionnément -> v_fraises=3 #item = à la folie -> v_fraises=4 #groupe = fraises, type=radio bouton, dispo=V	**Comment aimez vous les fraises ?** ○ un peu ○ beaucoup ○ passionnément ◉ à la folie

Un groupe de 4 radio boutons (type=radio bouton) est créé.
Les 4 items définissent ces radio boutons :

- texte du radio bouton
- action en cas de clic sur ce bouton → affectation de la variable ' v_fraises '

Les groupes de radio boutons se différencient des lignes de radio boutons par la possibilité d'une action différente pour chaque radio bouton.

- **#image = nomFichier.xxx**

Ex :

`#image = logo.jpg`	

Affiche une image qui peut illustrer un propos.

- **#addTexteAide**

Ajoute le texte de la balise <aide> dans le texte du formulaire.
Cela évite de recopier du texte.

Ex :

```
<aide>
Bonjour.
</aide>

<formulaire type=information>
#addTexteAide
Comment allez vous ?
</formulaire>
```

→ texte du formulaire =
Boujour.
Comment allez vous ?

8 – Les balises des formulaires

Dans un formulaire, les balises servent à mettre en forme du texte et à exécuter des scripts.

- **balise <?>**

Ex :

`<? Quel est votre poids ?>`	**Quel est votre poids ?**

La question est en gras et se démarque du reste du texte.

- **balise <notes>**

Ex :

`<notes>`
`oui -> retour au menu`
`</notes>`
	oui -> retour au menu

La note est en petits caractères

- **balise <style = style1, style2, ...>**

Ex :

`<style = bold, couleur=marron>`
`Test choisi :`
`</style>`
	Test choisi :

Le style du texte peut être totalement défini. Ici, texte gras de couleur marron.

Les styles (style1, style2) peuvent être :

italic	→	texte en italiques
bold	→	texte gras
small	→	texte en petits caractères
large	→	texte en grands caractères
couleur=nom	→	texte de la couleur choisie (1)
taille=12	→	texte de taille 12
police=nom	→	police de nom choisi (2)

(1) nom couleur : bleu, rouge, noir, gris, grisF, rose, orange, vert, vertF, cyan, magenta, jaune, marron violet

(2) nom police : Courier New, Arial, Comic Sans MS, Rockwell, SansSerif, Times New Roman, Verdana

- **balise <scripFormulaire>**

Permet d'exécuter un script en javaScript. Voir plus loin.

Cette balise doit être dans un formulaire

9 – Les actions des instructions des formulaires

Les actions contenues dans les instructions des formulaires sont exécutées :
- soit lors du clic sur le bouton Ok de la boite du formulaire.
- soit lors du clic sur un composant visuel du formulaire qui y répond.

Les actions des instructions font suite à la flèche -> (faite de 2 caractères)
Il peut y avoir plusieurs actions séparées par des virgules.

Les actions sont de 2 type :
- action d'affectation v_X
- actions de déplacement dans l'arbre end, goto_suite, goto_script, goto_branche(X)

Les actions de déplacement sont intéressantes à effectuer en temps réel, c'est à dire lors du clic sur un composant visuel.
Les composants visuels réactifs sont les groupes de boutons et les liens.
Un clic sur une zone d'édition ou un bouton radio ne provoque aucune réaction.
Les actions de déplacement sont donc à placer dans les groupes de boutons et les liens.

- **action : v_X**

Ex :

```
#inputTxt = poids -> v_poids
#inputRB = sexe(masculin,féminin) -> v_sexe
#inputCB = couleur(bleu,rouge,vert,jaune) -> v_couleur

#item=oui -> v_menu=oui, goto_script
#item=non -> v_menu=non
#groupe=goto, type=bouton

#item=oui -> v_chaleur=oui, goto_suite
#item=non -> v_chaleur=non, goto_suite
#groupe=temp, type=bouton
```

Les instructions permettent de créer des composants graphiques munis de noms.
Ces noms sont situés après le premier caractère = de l'instruction.
Noms correspondant aux exemples : poids, sexe, couleur, goto, temp
(#item n'est pas un composant graphique mais une information pour en construire un)

Ces noms doivent être uniques dans l'arbre car ils servent à établir un lien entre le composant graphique et la variable v_X.
Ainsi, la valeur saisie dans une zone d'édition peut être affectée à la variable correspondante.

- **action : end**

Ex :

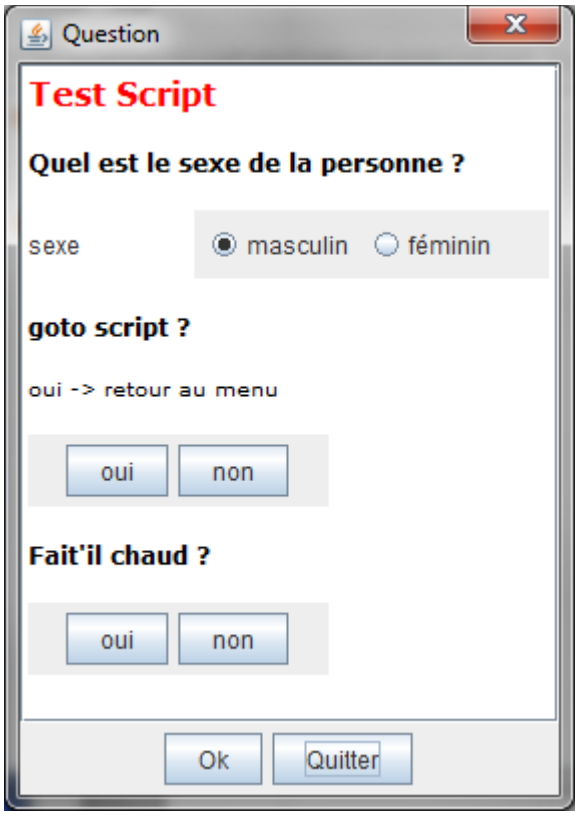
```
#lien = Quitter -> end
```

Un clic sur le lien 'Quitter' met fin au cheminement dans l'arbre.

- **action : goto_suite**

Ex :

```
<branche>
#nom=5
#titre=Test Script

<formulaire type=question>

<? Quel est le sexe de la personne ?>

#inputRB = sexe(masculin,féminin) ->
v_sexe

<? goto script ?>

<notes>
oui -> retour au menu
</notes>

#item=oui -> v_menu=oui, goto_script
#item=non -> v_menu=non
#groupe=goto, type=bouton

<? Fait'il chaud ?>

#item=oui -> v_chaleur=oui, goto_suite
#item=non -> v_chaleur=non, goto_suite
#groupe=temp, type=bouton
</formulaire>
```



Un clic sur le bouton oui ou non de la question fait'il chaud → on passe au formulaire suivant sans devoir cliquer sur le bouton Ok de la boite de dialogue (c'est plus intuitif)
Il est donc préférable de mettre les questions avec bouton en bas de formulaire.

- **action : goto_script**

goto_script provoque un saut dans le script **de fin de branche.**

Ex :

```
<scriptBranche>
  pAD.GotoBranche("0") ;
</scriptBranche>

</branche>
```

Ce script demande également un saut : vers la branche de nom '0'.

- **action : goto_branche(nomBranche)**

provoque un saut vers une branche de nom donné.

Ex :
```
#lien = Demande de données -> goto_branche(1)
#lien = Branchements -> goto_branche(2)
```

- **action : entree_se(symptome)**

Ex : entre le symptôme 'sodium = 142' dans le système expert.
```
#item = oui -> entree_se(sodium = 142)
```

10 – Les balises <scriptFormulaire> et <scriptBranche>

Un script **interne à l'arbre** est un code javaScript chargé d'exécuter un programme particulier.
Ce code doit donc respecter la syntaxe javaScript (cf plus loin)

Un script peut être placé :
1- en fin de branche : il est alors exécuté après les formulaires de la branche **<scriptBranche>**
2- dans un formulaire : il est exécuté pendant la création du formulaire **<scriptFormulaire>**

Un formulaire est une boite pour afficher des informations et poser des questions.
Cette boite est créée puis exécutée.
La création est l'insertion du texte et des éléments actifs : zones de saisie, boutons …
L'exécution affiche la boite et attend les actions de l'utilisateur : saisie, clic sur un bouton ...

Le scriptFormulaire est exécuté pendant la phase de construction de la boite pour une meilleure souplesse.
Ex : questions adaptées au sexe du patient au lieu de construire 2 branches fixes selon le sexe.
Ex : affichage de calculs selon des données saisies dans d'autres formulaires.

Cette balise ne doit pas contenir des instructions utilisant des données saisies dans le même formulaire car ces données ne seront disponibles qu'après la phase d'exécution qui se fera après la création.

incorrect	correct

`<branche>`
`<formulaire type=question>`
`<? Quel est votre nom ?>`
`#inputTxt=nom -> v_nom`
`<scriptFormulaire>`
`  pAD.Print("Bonjour " + v_nom) ;`
`</scriptFormulaire>`
`</formulaire>`
`</branche>`
	`<branche>`
	`<formulaire type=question>`
	`<? Quel est votre nom ?>`
	`#inputTxt=nom -> v_nom`
	`</formulaire>`
	`<formulaire type=information>`
	`<scriptFormulaire>`
	`  pAD.Print("Bonjour " + v_nom) ;`
	`</scriptFormulaire>`
	`</formulaire>`
	`</branche>`

- **balise <scriptFormulaire>**

Ex :

```
<formulaire type=information>
<scriptFormulaire>
  var LDL = v_cholT - v_HDL - (v_trigly / 5) ;
  var LDLConv = pAD.ConvertU(LDL, "LDL cholestérol") ;

  pAD.Print("LDL cholestérol = " + LDLConv) ;

</scriptFormulaire>
</formulaire>
```

Le script du formulaire récupère automatiquement les variables d'arbre de décision (variables v_xxx) créées au fil des formulaires. Ex : v_cholT, v_HDL, v_trigly.
Puis il calcule et affiche la valeur du LDL cholestérol.

- **balise <scriptBranche>**

Permet d'écrire un mini programme qui peut effectuer des calculs à partir de variables et orienter le chemin dans l'arbre.
Ce programme est exécuté en fin de branche, c'est à dire après l'exécution de tous les formulaires de la branche.
Les variables, créées dans les formulaires qui ont posé des questions, sont importées par le script et peuvent servir à divers calculs et actions.
Ce script peut également exporter des variables calculées localement et utilisables par l'arbre dans d'autres scripts.

Ex :

```
<branche>
#nom=1
#titre=Bilan lipidique

<formulaire type=question>

<? Quel est le taux de triglycérides ?>
#inputTxt = triglycérides -> v_trigly
</formulaire>

<scriptBranche>
  if (v_trigly>=4) pAD.GotoBranche("2B") ;
</scriptBranche>

</branche>
```

Le script de branche utilise la valeur de la variable de branche "v_trigly"
Si cette valeur est >= 4 le cheminement dans l'arbre est dérouté vers la branche "2B"

11 – La balise <include = nomInclude>

La balise <include> permet de stocker un texte qui pourra être injecté dans le texte d'un formulaire à l'aide d'une instruction de script ou d'une instruction de formulaire.
Intérêt des <include>

- réutiliser un même texte à plusieurs endroits.
- insérer un texte en fonction de conditions.

● injection de texte à l'aide d'un script : pAD.Include("xxx")

Ex :

```
<include = PECfrcv>
Une prise en charge des facteurs de risque cardiovasculaires est nécessaire.
</include>

<scriptFormulaire>
  ...
  if (cat>=1)
  {
    pAD.Includeln("PECfrcv") ;
  }
</scriptFormulaire>
```

L'exécution du script va insérer 1 ligne dans le texte du formulaire.

● injection de texte à l'aide d'une instruction : #include = xxx

Ex :

```
<include = EchecTrt>
En cas d'échec du traitement ou de rechute précoce du RGO :
 -> faire une fibroscopie oeso-gastro-duodénale.
 -> puis IPP à double dose.
</include>

<formulaire type = information>
Un IPP à pleine dose est recommandé pendant 4 semaines.

#include = EchecTrt

</formulaire>
```

12 – La balise <importation>

Cette balise sera décrite plus tard car elle permet une liaison avec un système plus complet de gestion des connaissances (SGC), comportant des systèmes experts, et qui n'est pas présenté ici.

13 – La balise <chargeSE4>

Charge un fichier de données structurées qui sera utilisé par l'arbre de décision.

Ex :

```
<chargeSE4>
PiluleContraceptive.txt
</chargeSE4>
```

Le fichier 'PiluleContraceptive.txt' est chargé temporairement en mémoire pour les besoins de l'arbre.
Ce fichier contient une liste des pilules contraceptives :

- nom de la pilule
- nombre de comprimés dans la plaquette
- nombre de comprimés inactifs

Des règles permettent de produire des données nouvelles à partir des données structurées.

Voir le chapitre : les données structurées.

14 – Les variables

L'arbre de décision gère 3 type de variables :

- les variables du système expert (variablesSE)
- les variables spécifiques à l'arbre (variablesAD)
- les variables locales spécifiques aux scripts JavaScript (variablesJS)

Ces 3 types de variables peuvent échanger leur valeur.

● les variables SE

Sont déclarées dans le dictionnaire du système expert.

Le lien variablesSE ↔ variablesAD est fait grâce au nom.

Ex : variable 'âge' du système expert

âge=55 ans	**Quel est l'âge de votre patient ?** âge (ans) 55,00

L'âge a été saisi dans le système expert → variableSE âge=55
Ensuite on exécute l'arbre de décision
	`<? Quel est l'âge de votre patient ?>`
	`#inputTxt = âge -> v_age`
	Quand l'arbre est exécuté la donnée âge est pré-saisie et la variableAD v_age vaudra 55,00

- **les variables AD**

Sont déclarées dans des instructions.

Ex :

```
#inputCB = couleur(bleu,rouge,vert,jaune) -> v_couleur
```

Déclare la variableAD 'v_couleur' reliée au composant graphique comboBox.
Un clic sur la valeur rouge du comboBox → v_couleur prend la valeur rouge.

Par convention les noms des variablesAD commencent par v_ et ne doivent pas contenir de lettre accentuée.
Ex de noms valides : v_sexe, v_antFam, v_diabete2

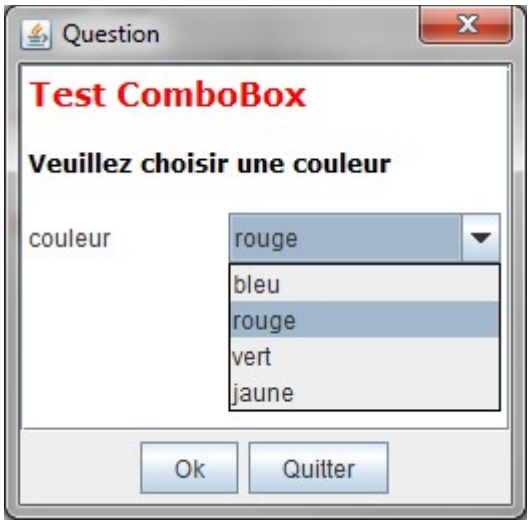
Ces variablesAD sont utilisables directement dans les scripts javaScript sans déclaration préalable.

Ex :

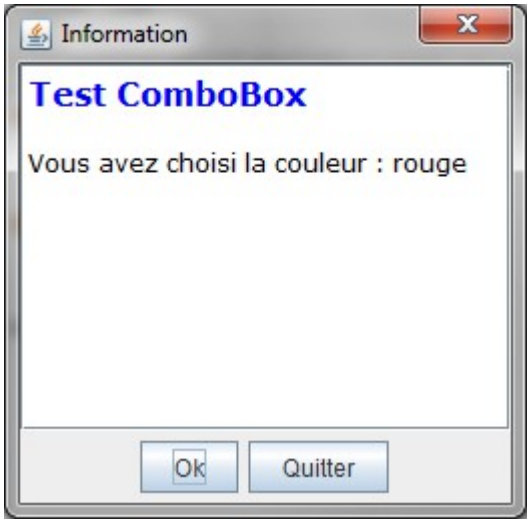
```
<scriptFormulaire>
  pAD.Print("Vous avez choisi la couleur : " + v_couleur) ;
</scriptFormulaire>
```

Exemple d'utilisation des variables

```
<branche>
#nom4
#titre=Test ComboBox

<formulaire type=question>

<? Veuillez choisir une couleur >

#inputCB = couleur(bleu,rouge,vert,jaune) ->
v_couleur
</formulaire>
```



```
<formulaire type=information>
<scriptFormulaire>
  pAD.Print("Vous avez choisi la couleur : " +
v_couleur) ;
</scriptFormulaire>
</formulaire>

</branche>
```



15 – Syntaxe javaScript

► Voyons l'exemple suivant :

```
<branche>
#nom=5
#titre=Test Script

<formulaire type=question>
<? Quel est le sexe de la personne ?>

#inputRB = sexe(masculin,féminin) -> v_sexe

<? Fait'il chaud ?>

#item=oui -> v_chaleur=oui, goto_suite
#item=non -> v_chaleur=non, goto_suite
#groupe=temp, type=bouton
</formulaire>

<formulaire type=information>
<scriptFormulaire>
  var offre   = "une petite glace ?\n" ;
  if (v_chaleur=="non") offre="un petit chocolat chaud ?\n" ;

  if (v_sexe=="masculin") pAD.Print("Bonjour monsieur " + offre) ;
  if (v_sexe=="féminin")  pAD.Print("Bonjour madame " + offre) ;
</scriptFormulaire>
</formulaire>

<scriptBranche>
  pAD.GotoBranche("0") ;
</scriptBranche>

</branche>
```

► Description du code exemple

Script du formulaire d'information (texte en gras)
(1) la variable locale javaScript ''offre'' est une chaine de caractères initialisée avec ''une petite glace ?''
cette variable n'est pas une variable d'arbre et peut ne pas commencer par v_
(2) si la variable v_chaleur est égale à non alors offre devient ''un petit chocolat chaud ?''
(3) si v_sexe est égal à masculin et v_chaleur égal à oui, affichage du texte ''Bonjour monsieur une petite glace ?''

Script de fin de branche
pAD.GotoBranche("0") provoque un saut à la branche 0.

► Syntaxe javaScript

Toute ligne de code doit se terminer par ;

Les variables javaScript sont déclarées par var nomVar
Ex : `var sexe="masculin") ;`

Les actions conditionnelles sont réalisées par :
```
if (condition) action ;
if (condition) { action1 ; action2 ; }
```

Le comparateur dans la condition est du type :
== pour tester l'égalité **(2 signes = et non 1 seul)**
!= pour tester la différence
les autres comparateurs sont classiques : >, >=, <, <=

Les chaînes de caractères doivent être entourées de guillemets.
Ex : `if (chaleur=="non") offre="un petit chocolat chaud ?\n" ;`
Le caractère \n provoque un passage à la ligne suivante.

Les nombres décimaux comportent des points et non des virgules.
Ex : 1.5 (correct) - 1,5 (incorrect)

opérateur =

Affecte une valeur à une variable :
ciel = "bleu" ;
age = 30 ;

opérateurs == >= <= !=

Testent 2 valeurs de même type

== teste l'égalité
>= teste supérieur ou égal
<= teste inférieur ou égal
!= teste la différence

opérateurs && ||

Ce sont les opérateurs logiques

&& teste le ET
|| teste le OU

Les instructions if, else

if permet de tester un condition. Si elle est vraie, le reste de la ligne est exécuté.
else est exécuté si la condition if est fausse. else n'est pas obligatoire.

if (température <= 10 && temperature <= 30) temperaturedouce="vrai" ;

!!! Noter que la condition à tester est située entre parenthèses

Attention : L'instruction return ne doit pas être utilisée dans les scripts.
Cette instruction est valide dans les scripts javaScript mais elle provoque un mauvais fonctionnement des scripts utilisés dans les arbres de décision.

16 – Instructions propres au système

Un script peut utiliser des instructions qui ne font pas partie de JavaScript mais qui y ont été ajoutées pour répondre aux besoins des AD.
Ces instructions, préfixées par "pAD." , sont les suivantes :

- **pAD.Print(''Texte'')**

```
pAD.Print("Bonjour") ;
```
Affiche le texte délimité par les guillemets → affiche Bonjour

```
pAD.Print("Bonjour monsieur " + v_nom) ;
```
Affiche ''Bonjour monsieur '' suivi de la valeur de la variable locale v_nom.

```
pAD.SetStyleTemporaire("bold") ;
pAD.Print("Bonjour.\n") ;
pAD.ResetStyleTemporaire() ;
```
Affiche Bonjour en gras et reprend un style d'affichage normal.

```
pAD.Print("\n") ;
```
Affiche une ligne vide

```
if (trt==true) pAD.Print("#lien = Traitement -> goto_branche(5B)") ;
```
Affiche un lien qui provoquera un saut à la branche de nom 5B (en cas de clic sur le lien Traitement)
Ce lien ne sera affiché que si la variable locale trt est égale à true.
Ceci est intéressant pour construire des menus dépendants du contexte.

- **pAD.Println("Texte")**

```
pAD.Println("Bonjour") ;
```

Affiche le texte Bonjour suivi d'un passage à la ligne suivante.

- **pAD.Includeln("Texte")**

```
pAD.Includeln("PrevDemain") ;
```

Cette instruction va inclure le texte contenu dans la balise <include = PrevDemain> et ajouter un passage à la ligne.

- **pAD.Export()**

Les variables d'un script javaScript et les variables d'un arbre sont différentes même si elles ont le même nom.
Un système permettant l'échange de données entre ces 2 groupes de variables est donc nécessaire.
Une variable créée dans un script peut être utilisée dans le formulaire d'une autre branche.

Ex :

```
 1 <branche>
 2 <scriptBranche>
 3   var total = v_nb1 + v_nb2 ;
 4   pAD.Export("total", String(total)) ;
 5 <scriptBranche>
 6 </branche>
 7
 8 <branche>
 9 <formulaire=information>
10 Voici le résultat demandé :
11 <scriptFormulaire>
12   pAD.Print("Total = " + total + " Km") ;
13 </scriptFormulaire>
14 </formulaire>
15 </branche>
```

Ligne 3 : calcul d'une somme :
- à partir des variables d'arbre de décision v_nb1 et v_nb2 (nom débutant par v_)
- stockée dans la variable locale "total" (nom ne débutant pas par v_)

Ligne 4 : la variable "total" est exportée dans l'arbre de décision.
!!! A noter : toutes les exportations doivent être faites sous forme de chaine de caractère (String).
la variable numérique "total" doit donc être convertie en chaine : instruction String()

Ligne 12 : affiche la valeur de la variable total (qui a été créée dans une branche différente)

Ex : nb1=8 et nb2=4

```
Voici le résultat demandé :
Total = 12 Km
```

- **pAD.SetStyleTemporaire(''Style'')**

```
pAD.SetStyleTemporaire("Bold") ;
```

Définit un style temporaire pour le texte à afficher. Style gras dans l'exemple.

- **pAD.ResetStyleTemporaire()**

Rétablit le style utilisé avant l'instruction SetStyleTemporaire(''Style'')

- **pAD.GotoBranche(''nomBranche'')**

```
pAD.GotoBranche("2") ;
```

Provoque un saut à la branche de nom "2" – Remarquez que 2 est entre guillemets.

- **pAD.isVar(''nomVar'')**

Ex : `if ( ! pAD.isVar("v_age")) ... ;`

Les variables de l'arbre de décision sont crées quand une donnée demandée est saisie.
Si la saisie n'est pas faite et que cette variable est utilisée dans un script cela provoque une erreur.

L'instruction de l'exemple précédent permet de tester l'existence de la variable v_age et donc de prévenir une erreur lors de son utilisation dans un script.

- **pAD.Exit()**

Provoque la fin de l'exécution de l'AD

Ex : `if ( ! pAD.isVar("v_age")) pAD.Exit() ;`

Si la variable v_age n'a pas été créée (car non saisie) l'exécution de l'arbre est interrompue.

17 – Les données structurées

- **les fichiers de données structurées**

Ces fichiers sont dans le répertoire : Data/SAM/BC/Onto

Ils décrivent et contiennent des données structurées qui seront utilisées par les AD.

Ex : Extrait du fichier PiluleContraceptive.txt

```
<metadata>
Date de création = 01/12/2011
```

```
Date de modification = 04/12/2011
Auteur = Dr HOUSE
</metadata>

<template=Pilule>
nom
nbComprimés
nbComprimésInactifs     ; défaut=0
</template>


<data>

#useTemplate = Pilule

pilule combinée (pilule oestro-progestative)
  ADEPAL/21
  CYCLEANE 20/21, CYCLEANE 30/21
  JASMINELLE/21, JASMINELLE CONTINU/28/4
  MELODIA/28/4, MERCILON/21, MINESSE/28/4

pilule microprogestative (pilule progestative)
  CERAZETTE/28
  MICROVAL/28

+ pilule combinée
| CERAZETTE
> tolérance oubli = 12

+ MICROVAL
> tolérance oubli = 3

</data>
```

La balise <metadata>

Contient des informations sur le fichier lui même.

La balise <template=nomTemplate>

Contient le définition d'une structure de données.

Dans l'exemple précédent, le template de nom 'Pilule' décrit des pilules contraceptives.
On souhaite donc créer des données contenant :

- un nom (le nom de la pilule)
- le nombre de comprimés dans la tablette
- le nombre de comprimés inactifs dans la tablette (avec une valeur par défaut égale à zéro)

Attention : le champ 'nom' est obligatoire dans un template. Cela permet de retrouver une donnée par son nom.

La balise <data>

Contient :

- des noms de famille pour classer les données
- les données structurées conforme au modèle (template)
- des règles produisant des données nouvelles

L'instruction `#useTemplate = Pilule` permet de spécifier le modèle utilisé.

Les noms de famille débutent la ligne sans blancs

Dans l'exemple précédent 'ADEPAL' fait partie de la famille 'pilule combinée'

Les données structurées débutent la ligne par 2 blancs

Les champs de chaque donnée structurés sont séparés par '/'
Les données situées sur une même ligne sont séparées par une virgule.

Les règles débutent par un signe particulier

+ pour si
| pour ou
> pour alors

Dans l'exemple on définit la tolérance à l'oubli selon la pilule, en utilisant les noms de famille et les noms de pilule.

```
Si pilule combinée
ou cerazette
alors tolérance à l'oubli = 12
```

● utilisation des données structurées dans les AD

① Chargement du fichier de données structurées.

```
<chargeSE4>
PiluleContraceptive.txt
</chargeSE4>
```

Le fichier 'PiluleContraceptive.txt' est chargé par SE4

SE4 est le système qui :
- charge les données structurées en mémoire
- gère les règles agissant sur ces données structurées
- est appelé par l'arbre de décision pour importer des données

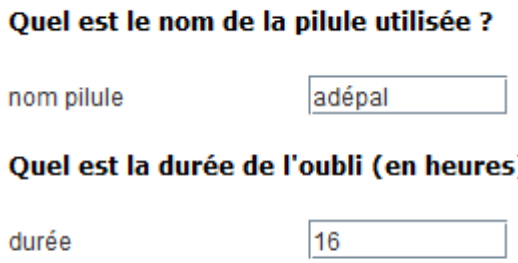
② Importation de données dans un script : Ex 1

```
<branche>
<formulaire type=question>

<? Quel est le nom de la pilule utilisée ?>

#inputTxt = "nom pilule" -> v_nomPilule #toSE4 [toMaj, delMot(GE)]
```

```
<? Quel est la durée de l'oubli (en heures) ?>

#inputTxt = "durée" -> v_dureeOubli

</formulaire>

<scriptBranche>
var toleranceOubli = pAD.ImportSE4("tolérance oubli") ;

if (v_dureeOubli <= toleranceOubli) pAD.GotoBranche("2") ;
  else pAD.GotoBranche("3") ;

</scriptBranche>

</branche>
```

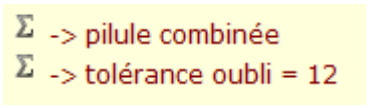
Etudions le code :

```
#inputTxt = "nom pilule" -> v_nomPilule #toSE4 [toMaj, delMot(GE)]
```



v_nomPilule → La variable d'arbre v_nomPillule prendra la valeur 'adépal'

#toSE4 → la valeur ADEPAL sera envoyée dans SE4 → déductions suivantes :



toMaj → la saisie est convertie en majuscules (→ ADEPAL) (1)

delMot(GE) → suppression du mot GE dans la saisie (2)

(1) (2) permet une correspondance avec les noms des données structurées qui sont en majuscule et sans les mots GE (génériques)

```
var toleranceOubli = pAD.ImportSE4("tolérance oubli") ;
```

la variable de script 'toleranceOubli' prend la valeur de la variable SE4 'tolerance oubli'

③ Importation de données structurées dans un script : Ex 2

```
<scriptFormulaire>
var nb              = pAD.ImportSE4("nbComprimés", v_nomPilule, "Pilule") ;
</scriptFormulaire>
```

Etudions le code ci dessus.

```
var nb              = pAD.ImportSE4("nbComprimés", v_nomPilule, "Pilule") ;
```

La variable de script 'nb' prend la valeur du champ 'nbComprimés' de la donnée structurée :

- correspondant au template 'Pilule'
- dont le nom est v_nomPilule c'est à dire 'ADEPAL'

→ nb = 21

Table des matières