Spécifications du mini-langage de description de taxons « LDT »

Version 4.0 – Sylvain Ard – 17/11/2023

Introduction

Ce langage permet aux experts de chaque observatoire du site observatoria.fr de décrire les taxons de son observatoire. On nomme ce langage LDT (langage de description de taxons). Ce qui se trouvera en italique gras sera à remplacer par des instructions quelconques.

Structure générale

Le texte en LDT de la description doit renvoyer au final du texte. Il est composé de texte brut qui s'affichera tel quel dans la description d'un taxon et de morceaux de code qui sont des instructions à exécuter. Ce code permet d'avoir de la description « dynamique » plutôt que « statique » avec le texte brut.

Pour séparer les codes proprement dits, on les encadre par des accolades ({ et }).

On appellera pour la suite ces morceaux « code » plutôt que la description complète du taxon et le texte brut « texte brut ».

Il y a 4 types de codes :

3 sont liés à une condition, ce sont le :

- {IF ***condition***} ou {IF ***condition*** THEN}
- {ELSE}
- {END IF}

Le premier est un IF (« si » en français), il est suivi d'une condition, si la condition est vraie ce qui suit est exécuté jusqu'au premier {ELSE} ou {END IF} suivant. Le ELSE est un « sinon » en français, le code entre lui et le premier {END IF} est exécuté si la condition du {IF} est fausse. Le {ELSE} est facultatif, s'il n'y en a pas le code entre le {IF} et le {END IF} sera exécuté seulement si la condition du {IF} est vraie.

Le quatrième type de code est {***instructions***}, on peut mettre ce qu'on veut comme instructions entre les accolades, elles seront affichées en texte.

A noter qu'on peut ajouter un « THEN » (« alors » en anglais) après l'instruction « IF » mais qu'il est facultatif. Il n'est là que pour une meilleure lisibilité.

Les types

Il y a 5 types dans LDT, les voici :

- Entier : un nombre entier.
- Flottant : un nombre entier ou à virgule.
- Chaîne de caractère : un texte quelconque, dans le code il est encadré par des guillemets doubles ("). Pour forcer un retour à la ligne dans une chaîne de caractère on écrira le code « \n » dans cette chaîne, pour forcer un guillemet double dans la chaîne dans le code on écrira « \ " », enfin pour écrire un backslash (« \ ») on écrira « \\ ». A noter que par mesure de sécurité toutes les autres balises HTML seront automatiquement supprimées. Ces codes sont valables aussi dans le texte brut à part le « \ " » qui peut s'écrire directement par « " ».
- Tableau : un ensemble ordonné de valeurs d'autres types (entier, chaîne de caractère, etc.), dans le code et dans son affichage par défaut le tableau commence par le caractère « [» et se termine par le caractère «] », les éléments sont séparés par des virgules.
- Booléen : soit « Vrai » soit « Faux », par exemple la condition est et doit être un booléen, s'il est seul dans un « code » il affichera « True » si vrai ou « False » si faux. Deux valeurs sont prédéfinies : la valeur « True » pour « vrai » et la valeur « False » pour « faux », vous pouvez les écrire tel quel dans le code.

Case unsensitive

Le langage LDT est « case unsensitive » c'est-à-dire que vous pouvez écrire les mots-clés (exemple : if, true, in, …) ainsi que les identificateurs (= noms de fonctions) indifféremment en minuscule ou en majuscule ou un mélange des deux. Exemple vous pouvez écrire indifféremment : getMetrics, GetMetrics, getmetrics, etc.

Les opérateurs

Il y a quatre types d'opérateurs :

- Les opérateurs arithmétiques
- Les opérateurs de comparaison
- Les opérateurs logiques
- Le « in »
- L'opérateur de concaténation

Nous allons les voir en détail.

Les opérateurs arithmétiques

Ils sont ceux que l'on apprend à l'école primaire (+,-,*,/), à noter que « * » est le multiplié et que « / » est le divisé.

A noter aussi que le * et le / ont une priorité plus grande que le + et le -.

Par exemple si on fait :

2+3*6 il faut calculer 2+(3*6) et non (2+3)*6, le * l'emporte sur le +, on dit qu'il a une priorité plus forte.

Les opérateurs de comparaison

Ce sont les >, le <, le <= (« inférieur ou égal »), le >= (supérieur ou égal), le =, le <> (« différent »), on les apprend aussi à l'école primaire, je ne vais donc pas ici redéfinir leur rôle, le résultat d'une opération avec ces opérateurs est un booléen, par exemple 5>3 vaut « vrai ».

Les opérateurs logiques

Ce sont les opérateurs de la « logique de Boole », c'est très simple vous verrez. Il y a le OR (« ou »), le AND (« et »), le XOR (« ou exclusif ») et le NOT (« non »).

Ils ont une priorité plus basse que les opérateurs arithmétiques et plus haute que les opérateurs logiques.

Voici un exemple simplifié :

{IF mesure1>mesure2 AND mesure3<=4}

Et voici les tableaux de calcul du résultat de ces opérateurs :

AND	Vrai	Faux
Vrai	Vrai	Faux
Faux	Faux	Faux

Pour lire ce tableau par exemple vous avez A AND B, A vaut vrai donc vous prenez la ligne « Vrai » à gauche et B vaut « Faux » donc vous prenez la colonne « Faux » en haut, vous croisez les deux et vous tombez sur le résultat à savoir « Faux ».

De la même façon voici les autres tableaux :

OR	Vrai	Faux
Vrai	Vrai	Vrai
Faux	Vrai	Faux

XOR	Vrai	Faux
Vrai	Faux	Vrai
Faux	Vrai	Faux

Le NOT est différent mais très simple, il inverse, donc NOT Vrai = Faux et NOT Faux = Vrai

Le « in »

Celui-ci renvoie « Vrai » si ce qui est à gauche (un entier, un flottant, etc) est contenu dans le tableau à sa droite.

Par exemple : {IF mesure1 IN [1,4,2]} signifie « si la mesure 1 vaut 1, 4 ou 2 ».

L'opérateur de concaténation « & »

Cet opérateur concatène deux chaînes de caractères c'est-à-dire qu'il les aboute, par exemple : "le"&"chien" donnera la chaîne « lechien ».

Les priorités

Chaque opérateur a une priorité, par exemple « 2+3 > 1 » sera exécuté comme cela « (2+3) > 1 » et non comme « 2+ (3>1) » car le « + » a une plus grande priorité que le >, donc il est « prioritaire » et est exécuté en priorité.

Voici les ordres de priorité des opérateurs de la plus basse à la plus haute :

- OR, XOR
- AND
- >, <, <=, >=, =, <>
- + , -, &
- *, /
- NOT
- IN
- (,)

A noter que l'on peut changer la priorité par défaut des opérateurs, par exemple pour forcer à faire (2+3)*5 on écrira tout simplement : « (2+3)*5 ». Les parenthèses peuvent s'imbriquer.

Les fonctions

Une fonction est comme en mathématique une sorte de machine qui prend des choses en entrée (les « paramètres ») et qui renvoie quelque chose en sortie (« la valeur de retour »). Par exemple en mathématique la fonction « sinus » prend en paramètre un angle et renvoie une valeur entre – 1 et 1.

A noter que la fonction a aussi un « nom » pour l'identifier, dans l'exemple précédent ce serait « sinus » et chaque paramètre ainsi que la valeur de retour a un type défini.

Prenons quelques fonctions du langage LDT comme exemple :

La fonction getMetrics(***nom_de_la_mesure***), renvoie un tableau de toutes les valeurs pour chaque spécimen de la mesure « nom_de_la mesure ». Nom de la mesure est le premier paramètre, il est de type « Chaîne de caractères », la valeur de renvoi est de type « tableau ».

Autre fonction : min(***tableau***), renvoie le minimum du tableau « tableau » en paramètre.

A noter que j'ai choisi de mettre les noms de fonction en anglais pour l'internationalisation d'observatoria.

Dernier exemple la fonction implode(***chaîne de caractère, tableau***) prend deux paramètres « chaîne de caractère » et « tableau », elle retourne une chaîne de caractère qui est les valeur de chaque case du tableau séparés par le paramètre « chaîne de caractère ».

Les paramètres sont aussi appelés « arguments », ils sont séparés par des virgules.

Annexe I : Inventaire des fonctions

Fonctions de tableaux

alwaysIn(arg1, arg2)

Renvoie « vrai » si toutes les cases du tableau « arg1 » valent « arg2 » sinon renvoie « Faux ». Si « arg2 » est un tableau, renvoie « vrai » si toutes les valeurs de « arg1 » sont parmi celles de « arg2 » et que parallèlement toutes les valeurs de « arg2 » sont dans « arg1 ». Evidemment elle renvoie « faux » sinon.

count(arg1**)**

Renvoie le nombre d'éléments du tableau « arg1 ».

getInterval(arg1)

Renvoie la chaîne « a-b » où « a » est le minimum du tableau « arg1 » et « b » le maximum du tableau « arg1 ».

C'est donc l'équivalent de {min(arg1)}-{max(arg1)}

implode (arg1, arg2)

Retourne une chaîne de caractère qui est les valeur de chaque case du tableau arg2 séparés par le paramètre « arg1 ». Exemples de valeurs pour « arg1 » : «
 » (retour à la ligne), « , », « ; »

implodeEnd (arg1, arg2, arg3)

Retourne une chaîne de caractère qui est les valeur de chaque case du tableau arg2 séparés par le paramètre « arg1 ». Exemples de valeurs pour « arg1 » : «
 » (retour à la ligne), « , », « ; ». La différence avec join est qu'entre le dernier et l'avant-dernier élément il y a « arg3 » et non « arg1 ». Exemple : join2(« , »,[« arrondi », « anguleux », « inconnu »], « ou ») donnera la chaîne de caractères :

« arrondi, anguleux ou inconnu ».

join (arg1,arg2)

Renvoie un tableau qui est le tableau « arg1 » abouté avec le tableau « arg2 ».

slice(arg1,arg2,arg3)

Renvoie le tableau contenant les valeurs du tableau « arg1 » de l'indice « arg2 » à l'indice « arg3 ». A noter que les indices commencent à 0 donc le dernier indice de « arg1 » est count(arg1)-1.

roundArray(arg1,arg2)

Arrondie toutes les valeurs du tableau « arg1 » au nombre de chiffres après la virgule « arg2 ».

sort (arg1,arg2,arg3)

Trie le tableau « arg1 » soit suivant l'ordre alphabétique (arg2=true) soit l'ordre numérique (arg2=false) et dans l'ordre croissant (arg3=true) ou décroissant (arg3=false).

unique(arg1)

Réduit le tableau « arg1 » à un tableau où chacune de ses valeurs n'est présente qu'en un seul exemplaire.

Fonctions de nombres

round(arg1,arg2)

Arrondit un nombre flottant « arg1 » au nombre de décimales « arg2 ».

Fonctions de mesures et de spécimens

A chaque fois qu'une mesure doit être passée en argument, c'est l'id de la mesure qui est attendu.

defined(arg1)

Renvoie « Vrai » si la mesure « arg1 » a plus d'une valeur pour au moins un spécimen dans ce taxon, équivalent à count(getMetric(arg1,-1))>0.

getBotanists()

Renvoie un tableau des botanistes ayant récolté ces spécimens.

getContributors()

Renvoie un tableau des botanistes ayant contribué à Observatoria, pour cette espèce dans l'ordre du nombre de spécimens renseignés.

getCountMetric(arg1)

Renvoie le nombre de mesures « arg1 » des spécimens du taxon en cours.

Equivalent de count(getMetrics(arg1,-1))

getCountMetricSpecimens(arg1)

Renvoie le nombre de spécimens du taxon en cours où la mesure « arg1 » est renseignée.

getCountries(arg1, arg2, arg3,arg4)

Renvoie un tableau des pays des spécimens dans l'ordre alphabétique, si arg1 (booléen) vaut « Vrai » alors à la suite de chaque pays se trouve le nombre de spécimens associés, arg2 est le texte à afficher avant le nombre de spécimens (donc après le pays) et arg3 est le texte à afficher après le nombre de spécimens. Arg4 est la langue, pour l'instant « en » ou « fr ».

getDepts(arg1,arg2,arg3)

Pour la France renvoie un tableau des départements des spécimens dans l'ordre alphabétique, si arg1 (booléen) vaut « Vrai » alors à la suite de chaque département se trouve le nombre de spécimens associés, arg2 est le texte à afficher avant le nombre de spécimens (donc après le département) et arg3 est le texte à afficher après le nombre de spécimens.

getMetrics(arg1,arg2)

Renvoie le tableau des valeurs de la mesure « arg1 » pour chaque spécimen du taxon en cours. « arg2 » est un paramètre qui indique le nombre de chiffres après la virgule d'arrondissement des valeurs, s'il vaut « -1 » aucun arrondi n'est fait.

getMetricsMin(arg1,arg2)

Renvoie un tableau de toutes les valeurs de la mesures « arg1 » qui sont les valeurs minimales de « arg1 » au sein de chaque individu. « arg2 » est un paramètre qui indique le nombre de chiffres après la virgule d'arrondissement des valeurs, s'il vaut « -1 » aucun arrondi n'est fait.

getMetricsMax(arg1,arg2)

Renvoie un tableau de toutes les valeurs de la mesures « arg1 » qui sont les valeurs maximales de « arg1 » au sein de chaque individu. « arg2 » est un paramètre qui indique le nombre de chiffres après la virgule d'arrondissement des valeurs, s'il vaut « -1 » aucun arrondi n'est fait.

getSources()

Retourne un tableau des sources (la plupart du temps herbiers) des spécimens.

getSpecimens()

Renvoie un des spécimens utilisés pour la description.

Fonctions statistiques

average(arg1)

Renvoie la moyenne des valeurs du tableau « arg1 ».

max (arg1)

Renvoie le maximum des valeurs du tableau « arg1 ».

median(arg1)

Renvoie la médiane des valeurs du tableau « arg1 ».

min (arg1)

Renvoie le minimum des valeurs du tableau « arg1 ».

standardDeviation(arg1)

Renvoie l'écart-type des valeurs du tableau « arg1 ».

Annexe 2 : Exemple de programme

Programme :

{IF defined('largeur des lobes terminaux')}

Feuilles à lobes de largeur variant, de {min(getMetrics(4,1))} à

{max(getMetrics(4,1))} cm

{ENDIF}

Résultat :

Feuilles à lobes de largeur variant, sur une même plante, de 5.1 à 7.2 cm