



Passerelle HTTP-LDAP

DOCUMENT DE REFERENCE

Version 3.20

SYSIUM

Octobre 2001

TABLE DES MATIERES

<u>1</u>	<u>INTRODUCTION</u>	7
<u>1.1</u>	<u>DOCUMENT ORDINAIRE</u>	7
<u>2</u>	<u>BALISES PAR GROUPE</u>	8
<u>2.1</u>	<u>TYPES DE REQUÊTE</u>	8
<u>2.1.1</u>	<u>Consultations</u>	8
<u>2.1.2</u>	<u>Mise à jour d'un DSA</u>	8
<u>2.1.3</u>	<u>Paramètres de requête</u>	8
<u>2.2</u>	<u>CHAMPS DE VALEUR</u>	9
<u>2.3</u>	<u>ENCHAÎNEMENT DES REQUÊTES</u>	10
<u>2.4</u>	<u>DÉTERMINATION DYNAMIQUE DES MODÈLES</u>	10
<u>2.5</u>	<u>STRUCTURES DE CONTRÔLE</u>	11
<u>2.6</u>	<u>MANIPULATION DES CHAÎNES DE CARACTÈRES</u>	11
<u>2.7</u>	<u>UTILISATION DES VARIABLES</u>	11
<u>2.8</u>	<u>MANIPULATION DES FICHIERS</u>	12
<u>2.9</u>	<u>ENVOI DES EMAILS</u>	12
<u>2.10</u>	<u>DIVERS</u>	12
<u>3</u>	<u>RÉFÉRENCES DES BALISES WXD</u>	14
<u>4</u>	<u>INDEX</u>	110

1 Introduction

Les balises sont entourées de "[]".

Les caractères composant une balise doivent être :

- soit tous en majuscules (ex: [AV]),
- soit tous en minuscules (ex: [/av]).

1.1 Document ordinaire

Un document "ordinaire" est un document ne commençant pas par une balise du type de requête (voir plus loin). Il n'y aura pas de requêtes X.500. Les balises devront être précédées du "\" pour apparaître dans le résultat final.

2 Balises par groupe

2.1 Types de requête

2.1.1 Consultations

[READ]	effectuer une lecture de l'entrée en récupérant les valeurs des attributs se trouvant dans le modèle.
[LIST]	en plus de la lecture de [READ], effectuer une recherche au niveau fille de l'entrée
[SEARCH]	en plus de la lecture de [READ], effectuer une recherche dans le sous-arbre de l'entrée
[COMPARE]	comparer les valeurs d'une entrée avec des valeurs fournies
[AUTH]	authentification, utilisé pour vérifier les mots de passe.

2.1.2 Mise à jour d'un DSA

[ADD]	ajout d'une entrée
[DELETE]	suppression d'une entrée
[MODIFY]	modification d'une entrée
[MODIFYRDN]	modification d'un RDN
[COPY]	copie d'une entrée

2.1.3 Paramètres de requête

[BASE]dn[/BASE] :
 nom canonique (*Distinguished Name*) de l'entrée de base des requêtes.

[FILTER]filtre[/FILTER] :
 filtre à utiliser pour les requêtes.

[FTAG]drapeau[/FTAG][FVAL]valeur[/FVAL] :
 drapeau et valeur pour la recherche des filtres dans un fichier de configuration.

[DEREF]x[/DEREF] :
 gestion des déréférences des alias.

[INPUT_ATTR]attribut=valeur[/INPUT_ATTR] :
 valeur d'attribut fixe à prendre en compte dans une opération de mise à jour.

[RETURN_ATTR]attribut1,attribut2[/RETURN_ATTR] :
 définition explicite des attributs à récupérer.

[LDAP_HOST]host[/LDAP_HOST] :
 adresse IP et le numéro de port du serveur LDAP. Utilisés pour diriger les requêtes LDAP vers un serveur différent de celui défini dans le fichier de configuration.

[TIMELIMIT]n[/TIMELIMIT] :
 :

[SIZELIMIT]n[/SIZELIMIT] :
 Limiter la recherche par le temps ou par le nombre d'entrées dans le résultat.

2.2 Champs de valeur

[DN modificateurs] :
 le nom canonique de l'entrée.

[AN] :
 le nom d'attribut.

[AV modificateurs]nom de l'attribut[/AV]:
la valeur de l'attribut dont le nom est spécifié.

[DAV modificateurs]...[/DAV]:
la valeur de l'attribut dont le nom est obtenu dynamiquement.

[ELI modificateurs]...[/ELI]:
liste des entrées.

[ALI modificateurs]...[/ALI]:
liste des attributs.

[AVLI NAME="nom de l'attribut"]...[V]...[V]...[/AVLI]:
liste des valeurs de l'attribut dont le nom est spécifié.

[DAVLI]...[LI][V]...[V]...[/DAVLI]:
liste des valeurs de l'attribut dont le nom est obtenu dynamiquement (avant [LI]).

[SORTBY]...[/SORTBY]:
tri d'une liste [ELI] ou [AVLI] par la valeur du champ.

[HEADER]...[/HEADER]:
en-tête d'une liste [ELI] ou [AVLI].

[FOOTER]...[/FOOTER]:
pied de page d'une liste [ELI] ou [AVLI].

2.3 Enchaînement des requêtes

[INC modificateurs]réf. modèle[/INC]:
inclusion d'un modèle.

[INLINE modificateurs][PARAM]...[/PARAM]...[/INLINE]:
inclusion d'un modèle inline.

2.4 Détermination dynamique des modèles

[EMDL modificateurs]...[/EMDL]:
nom du modèle correspondant à l'entrée dont le DN est le résultat d'interprétation du contenu du champ.

[AMDL]...[/AMDL] :
modèle correspondant à un attribut dont le nom est le résultat
d'interprétation du contenu du champ.

2.5 Structures de contrôle

[IF]conditions[THEN]...[ELSE]...[/IF] :
prendre en compte le contenu du champ si l'entrée satisfait aux
conditions.

[WHILE]conditions[DO]...[/WHILE]:
boucle tant que les conditions sont vraies.

2.6 Manipulation des chaînes de caractères

[GETBEFORE PAT=str]...[/GETBEFORE]:
retourne la partie du champ avant « str ».

[GETAFTER PAT=str]...[/GETAFTER]:
retourne la partie du champ après « str ».

[STRLEN]...[/STRLEN]:
retourne la longueur de la chaîne.

[SUBSTR]...[/SUBSTR]:
extraire une sous-chaîne de caractères.

[SUBINDEX]...[/SUBINDEX]:
retourne la position d'une sous-chaîne.

2.7 Utilisation des variables

[SET_MVAR]var=...[/SET_MVAR] ou

[SV]var=...[/SV]:
définir une variable "var".

[GET_MVAR]var[/GET_MVAR] ou

[GV]var[/GV]:
récupère la valeur d'une variable "var".

2.8 Manipulation des fichiers

[SAVE][PATH]...[/PATH]...[/SAVE]:
sauvegarde le contenu dans un fichier (sur le serveur).

[REMOVE]...[/REMOVE]:
supprimer un fichier (sur le serveur).

2.9 Envoi des emails

[SENDMAIL][TO]...[FROM]...[SUBJECT]...[MSG]...
[MSGFILE]...[ATTACH]...[/ATTACH]
[ATTACHFILE]...[/ATTACHFILE] [/SENDMAIL]:

envoyer un email par le serveur.

2.10 Divers

[ENC]...[/ENC]:
encoder les caractères spéciaux dans le champ pour l'utiliser
dans une URL.

[DECODE]...[/DECODE]:
l'opération inverse de [ENC]...[/ENC].

[ARG]...[/ARG]:
obtenir la valeur d'un argument d'une requête HTTP.

[ENV]...[/ENV]:
obtenir la valeur d'une variable d'environnement sur le
serveur.

[CUR_DIR]:
récupère le host et le port du serveur LDAP courant .

[SERVER_TIME] :
récupère la date courant du serveur .

[NB_CHILDREN] :
nombre d'entrées filles.

[SIZE_LIMIT_EXCEEDED] :
indicateur de dépassement de la limite de taille pour une
requête LDAP.

[CR] :
génère un retour chariot ("\n").

[ON_ERROR] . . . [/ON_ERROR] :
traitement des erreurs.

[REM] . . . [/REM] :
Commentaire.

3 Références des Balises WXD

Ce chapitre contient la description des balises WXD

[ADD]

Syntaxe

[ADD]

[INPUT_ATTR] *attribut = valeur de l'attribut*[/INPUT_ATTR]

Paramètres

en paramètre d'un modèle :

DN=DN_entrée : dn de l'entrée à créer, paramètre obligatoire.

ATTRNAME =nom : nom de l'attribut à créer.

ATTRVAL=valeur : valeur de l'attribut à créer.

Description

Permet de créer une entrée dans un annuaire.

Deux possibilités :

Si on sait qu'un modèle sera toujours utilisé pour effectuer des créations avec des valeurs et des attributs fixes, on utilise les balises [INPUT_ATTR]...[/INPUT_ATTR].

Si on veut une création avec des attributs variables, on peut utiliser un modèle contenant seulement le balise [ADD] et appeler celui-ci avec des paramètres qui peuvent être fournis soit dans l'URL, soit dans les champs d'un formulaire HTML et transmis par les méthodes GET ou POST de la commande *submit* de HTML.

Dans la suite des ATTRNAME et ATTRVAL, un nom d'attribut est actif jusqu'à l'attribut suivant. On peut donc fournir plusieurs valeurs à la suite d'un nom d'attribut.

D'autres arguments peuvent aussi venir s'intercaler entre ATTRNAME et ATTRVAL.

Tout ATTRVAL non précédé de ATTRNAME est ignoré.

Tout ATTRVAL précédé d'un ATTRNAME = avec le nom vide est aussi ignoré.

L'ordre des paramètres est important, car on attend la valeur d'un attribut après son nom.

Exemples

Création d'une entrée de l'annuaire avec des attributs en dur :

```
[ADD]
[BASE]cn=durant, ou=personnel, o=calice group, c=fr[/BASE]
[ INPUT_ATTR]objectClass=top[/INPUT_ATTR]
[ INPUT_ATTR]objectClass=person[/INPUT_ATTR]
[ INPUT_ATTR]cn=Paul Durant[/INPUT_ATTR]
[ INPUT_ATTR]sn=Durant[/INPUT_ATTR]
```

Création par un passage de paramètres à un modèle :

Si le modèle ne contient que la balise [ADD]

modele.html

```
[ADD]
```

On peut effectuer les créations en appelant le modèle avec les données en paramètres

```
http://host/model.html?DN=cn=durant,ou=personnel, o=calice groupe,
c=fr&ATTRNAME=objectClass&ATTRVAL=top&ATTRVAL=person&
ATTRNAME=cn&ATTRVAL= Paul Durant& ATTRNAME=sn&ATTRVAL= Durant
```

soit en utilisant un formulaire

```
[READ]
<HTML>
..
<FORM method="POST" action="model.html">
  <input type="hidden" name="DN" value="[DN]">
  <input type="hidden" name="ATTRNAME" value="objectClass">
  <input type="hidden" name="ATTRVAL" value="top">
  <input type="hidden" name="ATTRVAL" value="person">

  <input type="hidden" name="ATTRNAME" value="cn">
  Nom et prenom:
  <input type="text" name="ATTRVAL"><BR>

  <input type="hidden" name="ATTRNAME" value="sn">
  Nom :
  <input type="text" name="ATTRVAL"><BR>

  <input type="submit" value="OK">
</FORM>
</HTML>
```

[ALI]...[/ALI]

Syntaxe

[ALI AFLT=filtre1 AFLT=filtre2 ...]...[/ALI]

Paramètres

AFLT=valeur: applicable uniquement dans le champ [ALI AFLT=valeur]...[/ALI], il filtre les attributs qui correspondent à *valeur* qui est une expression régulière, précédée éventuellement par un "!" pour la négation. Ce paramètre peut apparaître plusieurs fois, chaque filtre étant appliqué dans l'ordre, seuls les attributs qui passent tous les filtres étant retournés.

Description

Liste les attributs d'une entrée.

Exemples

Lister tous les noms des attributs:

```
[ALI]  
Nom : [AN]<br>  
[/ALI]
```

Lister tous les attributs dont le nom commence par "acl" :

```
[ALI AFLT=acl.*]  
Nom : [AN]<br>  
[/ALI]
```

Lister les attributs acl et masterDSA :

```
[ALI AFLT=\\(\\(acl\\)|\\(masterDSA\\))]  
Nom : [AN]<br>  
[/ALI]
```

Lister tous les attributs différents de acl et masterDSA:

```
[ALI AFLT=!\\(\\(acl\\)|\\(masterDSA\\))]  
Nom : [AN]<br>  
[/ALI]
```

[AN]

Syntaxe

[AN]

Paramètres

Néant

Description

Affiche le nom de l'attribut courant dans un [ALI]... [/ALI]

Exemples

```
[READ]
[ALI]
  Nom de l'attribut : [AN]<br>
  Valeur de l'attribut : [AV]*[/AV]<br>
[/ALI]
```

[ARG]...[/ARG]

Syntaxe

[ARG mod]arguments [/ARG]

Paramètres

mod peut prendre les valeurs suivantes :

RDN=n	le même que celui de [DN], applicable quand la valeur de l'attribut est un DN.
UP=n	le même que celui de [DN], applicable quand la valeur de l'attribut est un DN.
UF	le même que celui de [DN], applicable quand la valeur de l'attribut est un DN.
UFN	retourner une valeur sous la forme " <i>User Friendly</i> " en se basant sur le fichier <i>friendlyfile</i> .
REPLACE="str1 str2"	Remplacer les «str1» par la chaîne «str2». Si les deux chaînes elles-mêmes contiennent des espaces, les espaces peuvent être exprimées par "\s", un retour chariot par un "\n", un linefeed par un "\r".
RE_REPLACE="/str1/str2/"	Remplacer les «str1» par la chaîne «str2». A la différence de REPLACE, «str1 » peut être une expression régulière. On peut utiliser un autre caractère que «/» pour séparer «str1 » et «str2 » (ex : #str1#str2#). Ceci est utile si «str1 » ou «str2 » contiennent elles-mêmes des «/».
UTCTIME	le temps est au format aammjjhhmmssZ (syntaxe d'attribut UTCTime dans X.500), le convertit en forme plus lisible comme "15 Novembre 1994 16:35:01 GMT".
URL	retourne la partie URL d'un attribut ayant une syntaxe labeledURL définie dans X.500.

URL_DESC	retourne la partie description d'un attribut ayant une syntaxe <code>labeledURL</code> définie dans X.500.
BEFORE=pattern	retourne la partie avant pattern. Ex : si la valeur est Service & Département, le modificateur <code>before="&"</code> permet d'obtenir Service.
AFTER=pattern	retourne la partie après pattern. Avec le même exemple que pour before, <code>after="&"</code> donne Département
FORMAT=<format>	formate le retour d'une valeur. Pour une valeur numérique, si on veut obtenir une chaîne de 4 caractères complétée éventuellement par des "0" devant, on peut utiliser <code>"% . 4d"</code> . [AV FORMAT= " % . 4d "] numero [/AV] va produire: 0012 si la valeur du champ numéro est 12. Pour obtenir une chaîne minimum de 19 caractères complétée par des espaces, on utilise <code>"%-19s"</code> , ou <code>"%+19s"</code> , "-" pour des espaces à la fin, "+" pour des espaces au début. Le format est celui de « printf ».
ASCII	convertit les caractères dont le code ASCII est supérieur à 128 en leur équivalent dans un code ASCII inférieur à 128.
FNAME	recupère le nom du fichier envoyé par un formulaire HTML.

Description

Permet de récupérer des arguments dans les modèles. Les arguments peuvent être passés par les méthodes POST ou GET de HTTP.

Exemple

Avec une méthode GET

```
http://machine/model.html?toto=1&titi=2
```

On récupère ces arguments dans le modèle avec `[ARG]...[/ARG]`

```
[ARG]toto[/ARG],  
[ARG]titi[/ARG]
```

ce qui donne

```
1,2
```

On peut mettre d'autres balises entre les balises [ARG]...[/ARG] par exemple si l'attribut *var* contient toto [ARG][AV]var[/AV][[/ARG] donne 1.

Exemple de chargement (upload) d'un fichier

Formulaire HTML :

```
<form action="model.html" enctype="multipart/form-data" method=post>  
<input type=file name="file1">  
<input type=submit>  
</form>
```

Si l'utilisateur a sélectionné un fichier "c:\photo\jean.jpg" pour le champ "file1",

Fichier model.html :

```
[ARG FNAME]file1[/ARG] : donne le nom du fichier "jean.jpg"  
[ARG]file1[/ARG] : donne le contenu du fichier "jean.jpg"  
[SAVE][PATH]/tmp/[ARG FNAME]file1[/ARG][[/PATH]  
[ARG]file1[/ARG][[/SAVE] : sauvegarde le contenu du fichier  
"jean.jpg" dans le fichier "/tmp/jean.jpg" sur le serveur.
```

Charger un fichier directement dans un attribut

Formulaire HTML :

```
<form action="model.html" enctype="multipart/form-data" method=post>
<input type=text name=DN>
<input type=hidden name=ATTRNAME value=jpegPhoto>
<input type=file name=ATTRVAL>
<input type=submit>
</form>
```

Fichier model.html :

```
[MODIFY]
```

C'est exemple montre le chargement d'une photo, mais on peut l'utiliser pour charger n'import quel type de fichier dans un attribut.

[AUTH]

Syntaxe

[AUTH]

Paramètres

en paramètre du modèle:

model.html?DN=<DN>&PASSWORD=<password>

Description

Modèle model.html:

```
[AUTH]
```

Le modèle renvoie 1 si l'authentification est réussie, 0 dans le cas contraire.

On peut l'utiliser pour comparer un mot de passe :

```
[ IF ] " [ INC RES ] model.html?DN=[ DN ] &
PASSWORD=[ ARG ] password [ / ARG ] [ / INC ] " [ THEN ]
    Mot de passe correct
[ ELSE ]
    Mot de passe invalid
[ / IF ]
```

On peut lui passer, comme pour les autres modèles, les arguments LDAP_HOST, LDAP_PORT, si l'authentification doit se faire dans un annuaire spécifique.

[AV]nom_attribut[/AV]

Syntaxe

[AV mod1 mod2 ...]nom_attribut[/AV]

Description

mod peut prendre les valeurs suivantes :

RDN=n	le même que celui de [DN], applicable quand la valeur de l'attribut est un DN.
UP=n	le même que celui de [DN], applicable quand la valeur de l'attribut est un DN.
UF	le même que celui de [DN], applicable quand la valeur de l'attribut est un DN.
UFN	retourner une valeur sous la forme " <i>User Friendly</i> " en se basant sur le fichier <i>friendlyfile</i> .
REPLACE="str1 str2"	Remplacer les «str1» par la chaîne «str2». Si les deux chaînes elles-mêmes contiennent des espaces, les espaces peuvent être exprimées par "\s", un retour chariot par un "\n", un linefeed par un "\r".
RE_REPLACE="/str1/str2/"	Remplacer les «str1» par la chaîne «str2». A la différence de REPLACE, «str1» peut être une expression régulière. On peut utiliser un autre caractère que «/» pour séparer «str1» et «str2» (ex : #str1#str2#). Ceci est utile si «str1» ou «str2» contiennent elles-mêmes des «/».
LG	retourner la valeur la plus longue d'un attribut multi-valué.
ST	retourner la valeur la plus courte d'un attribut multi-valué.
BIN="format facteur" :	la valeur de l'attribut représente des données binaires. Ces données sont alors converties en <i>format</i> avec un <i>facteur</i> de réduction, puis stockées dans un fichier temporaire. C'est l'URL du fichier qui est retournée. Si <i>format</i> est absent, pas

de conversion, si facteur absent, pas de réduction. Les deux paramètres ne peuvent pas être absents en même temps. Utilisable aussi bien pour l'image que pour le son à condition d'avoir spécifié les programmes de conversion correspondants dans le fichier de configuration (CONV_PGM, SCALE_PGM et CONV_SCALE_PGM).

UTCTIME	le temps est en format aammjjhmmssZ (syntaxe d'attribut UTCTime dans X.500), le convertit en forme plus lisible comme "15 Novembre 1994 16:35:01 GMT".
URL	retourne la partie URL d'un attribut ayant une syntaxe labeledURL définie dans X.500.
URL_DESC	retourne la partie description d'un attribut ayant une syntaxe labeledURL définie dans X.500.
BEFORE=pattern	retourne la partie avant pattern. Ex : si la valeur est Service & Département, le modificateur before="&" permet d'obtenir Service.
AFTER=pattern	retourne la partie après pattern. Avec le même exemple que pour before, after="&" donne Département
FORMAT=<format>	formate le retour d'une valeur. Pour une valeur numérique, si on veut obtenir une chaîne de 4 caractères complétée éventuellement par des "0" devant, on peut utiliser "% . 4d". [AV FORMAT=" % . 4d "]numero[/AV] va produire: 0012 si la valeur du champ numéro est 12. Pour obtenir une chaîne minimum de 19 caractères complétée par des espaces, on utilise "%-19s", ou "%+19s", "-" pour des espaces à la fin, "+" pour des espaces au début. Le format est celui de « printf ».
INDEX="n"	Permet de sélectionner une valeur donnée d'un attribut multivalué. [AV INDEX="1"]seeAlso[/AV] retourne la première valeur de la liste.

ASCII	converti les caractères (sans accent) dont le code ASCII est supérieur a 128 en leur équivalent dans un code ASCII inférieur a 128.
ISO8859	converti les caractères T61 en ISO 8859, avec accent.
T61	converti les caractères ISO 8859 (avec accent) en T61.
DUMP	Le modificateur DUMP indique qu'il faut retourner la valeur sans aucune conversion. Ce modificateur est utile pour renvoyer des valeurs binaires comme le son ou l'image.
NB_VAL	retourne le nombre de valeurs dans un attribut. [AV NB_VAL]member[/AV] [AVLI NAME=member]...[V NB_VAL]...[/AVLI]
LEN	applicable si la valeur est un DN, il retourne la partie du DN ayant n RDN à partir de la racine. Si n dépasse le nombre de RDN dans le DN, le DN entier est retourné..
NB_RDN	retourne le nombre de RDN si la valeur est un DN.

Description

Balise permettant de récupérer la valeur d'un attribut.

nom_attribut peut être un "*" si [AV]nom_attribut[/AV] se trouve dans [ALI]...[/ALI], dans ce cas, le nom de l'attribut est déterminé par l'attribut courant de la liste d'attribut [ALI]...[/ALI].

Remarquons qu'il faut donner "en dur" le nom de l'attribut dans [AV]...[/AV].

Une variante est [DAV]...[/DAV] où le nom de l'attribut est le résultat d'évaluation de ce qui est entre la balise, ce qui permet de récupérer dynamiquement le nom de l'attribut, par exemple dans un argument ([DAV][ARG]...[/ARG][DAV]) ou dans la valeur d'un autre attribut ([DAV][AV]...[/AV][DAV]).

Exemple

[AV]telephoneNumber [/AV]

Voir aussi

[DAV]...[/DAV], [AVLI]...[/AVLI], [DAVLI]...[/DAVLI]

[AVLI]...[/AVLI]

Syntaxe

[AVLI *parametre*="valeur"..[V]...[/AVLI]

Paramètres

Les paramètres peuvent être les suivants :

NAME="nom d'un attribut"

Nom de l'attribut dont on désire la liste des valeurs, ou « * »
lorsque l'on est dans une boucle [ALI].
Ce paramètre est obligatoire.

SEP="separateur" Ajoute un séparateur entre les valeurs d'un attribut listé.

[V *modificateurs*] récupère la valeur de l'attribut. Les modificateurs sont
identiques aux modificateurs de la balise [AV]...[/AV].

Description

Liste les valeurs d'un attribut multivalué.

Exemple

Utilisation avec le paramètre NAME :

```
[AVLI NAME="telephoneNumber" ]  
...  
[SORTBY reverse hidden]...[/SORTBY]  
...  
[V][ /AVLI ]
```

Utilisation avec le paramètre (séparateur) SEP :

```
[AVLI name="tel" SEP=" , " ]  
item  
[AVLI ]
```

va produire, si on a trois valeurs : item,item,item

[BASE]...[/BASE]

Syntaxe

[BASE] DN [/BASE]

Description

[BASE] DN [/BASE], où DN est le nom d'annuaire (DN) de base, dans sa forme UFN, pour la recherche (il indique le nom du nœud de l'arbre LDAP/X.500 servant de point de départ à la recherche).

Remarque

Le tag [BASE]...[/BASE] n'est plus obligatoire dans un modèle.

La raison est que beaucoup de modèles sont toujours utilisés avec un DN en paramètre.

Si jamais un modèle sans [BASE]...[/BASE] est utilisé sans DN=..., une erreur est signalée.

[CGI_ENV]...[/CGI_ENV]

Syntaxe

[CGI_ENV]variable d'environnement CGI[/CGI_ENV]

SERVER_SOFTWARE	The server software
SERVER_NAME	The server hostname
GATEWAY_INTERFACE	The CGI specification revision
SERVER_PROTOCOL	The name and revision of info protocol
SERVER_PORT	The port number for the server
REQUEST_METHOD	The info request method
PATH_INFO	The extra path info
PATH_TRANSLATED	The translated PATH_INFO
DOCUMENT_ROOT	The server document root directory
SCRIPT_NAME	The script name
QUERY_STRING	The query string (FORM GET)
REMOTE_HOST	The hostname making the request
REMOTE_ADDR	The IP address of the remote host
AUTH_TYPE	The authenticated method
REMOTE_USER	The authenticated user
REMOTE_IDENT	The remote user
CONTENT_TYPE	The content type of data (POST, PUT)
CONTENT_LENGTH	The length of the content
HTTP_ACCEPT	The MIME types that the client will accept
HTTP_USER_AGENT	The browser of the client
HTTP_REFERER	The URL of the referer

Description

Renvoie la valeur de la variable d'environnement spécifiée entre les tags.

La liste des variables sont celle utilisées dans l'API CGI standard.

Exemple

```
[CGI_ENV]REMOTE_ADDR[/CGI_ENV]
```

donne l'adresse IP du client Web.

[COMPARE]

Description

Comparaison des valeurs d'attributs.

On appelle le modèle avec :

```
http://host:port/model.html?DN=DN_entree&ATTRNAME=attr1&ATTRVAL=val11&ATTRVAL=val12&ATTRNAME=attr2&ATTRVAL=val21
```

On compare les valeurs des attributs de l'entrée DN_entree avec les valeurs fournies en paramètre, si toutes les valeurs de l'entrée sont égales aux valeurs fournies, le modèle affiche "1", sinon "0".

Notons qu'il peut y avoir plusieurs ATTRVAL après un ATTRNAME.

On utilise cette opération en général dans un sous modèle:

```
[IF][INC RES]compare.html?DN=<DN_entree>&
ATTRNAME=<nom_attr>& ATTRVAL=<val_attr>[/INC][THEN]
  [REM]traitement quand c'est égal[/REM]
[ELSE]
  [REM]traitement quand ce n'est pas égal[/REM]
[/IF]
```

Si certaines valeurs doivent être toujours vérifiées, on peut les mettre en dure dans le modèle (comme pour la modification):

```
[COMPARE]
[INPUT_ATTR]objectclass=person[/INPUT_ATTR]
```

[COND]...[/COND] (obsolète)

Syntaxe

**[IF][COND]expression[/COND]...[/IF]
[IF][COND]expression[/COND]...[ELSE]...[/IF]**

Cette syntaxe obsolète, elle est remplacée par :

**[IF]expression[THEN]...[/IF]
[IF]expression[THEN]...[ELSE]...[/IF]**

Les expressions restent les mêmes.

Paramètres

Les conditions au format :

expression ⇔ l'expression doit s'évaluer en vrai ou faux
expression & expression ⇔ ET logique
expression | expression ⇔ OU logique
!(expression)

On peut combiner les expressions ci-dessus :

(expression & (expression | !(expression)))

Les expressions peuvent être égales à:

0 ⇔ Faux
1 ⇔ Vrai

ou du type :

chiffre *opérateur* chiffre

"chaîne" *opérateur* "chaîne"

Avec les opérateur suivants :

== ⇔ égale
!= ⇔ différent de
> ⇔ supérieur a
< ⇔ inférieur a
>= ⇔ supérieur et égale à

Avec les opérateur suivants:

== ⇔ égale
!= ⇔ diffèrent de
> ⇔ supérieur a
< ⇔ inférieur a
>= ⇔ supérieur et égale à

<= ⇨ inférieur et égale à

Chiffre peut être le résultat d'un calcul :

(1 + 2)

(3 - 1)

([AV]salaire[/AV] * 12)

([NB_CHILDREN] / 3)

<= ⇨ inférieur et égale à

~ ⇨ comparaison avec une expression régulière.

"" entourant les opérands qui indiquent que l'on est dans une comparaison de chaîne de caractères.

Description

Permet de vérifier des conditions sur des entrées d'un annuaire.

Exemple

Comparaison de chaînes de caractères

```
[ IF ] [ COND ] " [ AV ] telephoneNumber [ / AV ] " == " 26.98.76.89 " [ / COND ]  
...  
[ / IF ]
```

Comparaison avec une expression régulière et de valeur numérique:

```
[ IF ]  
[ COND ] ( " [ DN ] " ~ " .*InfoPointu" ) & ( [ AV ] age [ / AV ] == 25 ) [ / COND ]  
...  
[ / IF ]
```

[COPY]

Syntaxe

[COPY]

En paramètre d'un modèle :

DN=*dn de l'entrée copiée*

NEWDN=*dn cible* ,

Paramètres

Utilisé en paramètre d'un modèle :

DN=DN_entrée

DN de l'entrée à copier

NEWDN=nouveau_DN

nouveau DN de l'entrée.

Description

Effectue la copie d'une entrée, tous les attributs sont copiés.

Exemple

copy.html :

[COPY]

L'appel du modèle suivant effectue la copie du nœud dn1 en dn2 (toutes les valeurs et tous les attributs sont copiés).

copy.html?DN=dn1&NEWDN=dn2

[CR]

Syntaxe

[CR]

Description

Force un saut de ligne là où ils ont l'habitude d'être ignoré.

Exemple

```
[ELI]  
[AV]telephoneNumber[/AV]  
[/ELI]
```

Va générer des valeurs les unes après les autres

```
8989 5656 5656 2323
```

Avec [CR] :

```
[ELI]  
[AV]telephoneNumber[/AV][CR]  
[/ELI]
```

WXD génère un numéro de téléphone par ligne.

```
8989  
5656  
5656  
2323
```

HTML ignorant les retours chariot, cette balise est notamment utile pour des modèles en format texte.

[CUR_DIR]

Syntaxe

[CUR_DIR]

Description

Raccourci pour :

```
"LDAP_HOST=[ARG]LDAP_HOST[/ARG]&LDAP_PORT=[ARG]LDAP_PORT[/ARG]"
```

Si un sous modèle ou un lien hypertexte contenu dans un modèle utilise le même serveur LDAP, au lieu de faire :

```
model.html?LDAP_HOST=[ARG]LDAP_HOST[/ARG]&  
LDAP_PORT=[ARG]LDAP_PORT[/ARG]
```

On fait :

```
model.html?[CUR_DIR]
```

Pour une inclusion ([INC]...[/INC] y compris [INC RES]...[/INC]), en absence d'indication explicite, on considère que le sous modèle utilise le même serveur que le modèle parent, [CUR_DIR] n'est donc pas nécessaire.

Voir aussi

[LDAP_HOST]...[/LDAP_HOST], [LDAP_PORT]...[/LDAP_PORT]

[DAV]...[/DAV]

Syntaxe

[DAV mod1 mod2 . . .]...[/DAV]

Description

Cette balise a exactement la même syntaxe que [AV]...[/AV].

La seule différence est que dans [AV]...[/AV], le nom de l'attribut doit être spécifié "en dur", par exemple [AV]cn[/AV], alors que [DAV]...[/DAV] accepte d'autres balises pour obtenir dynamiquement le nom de l'attribut (d'où le nom DAV pour Dynamic Attribut Value).

Cette balise est surtout utile pour créer des modèles d'inclusion utilisables pour des attributs dont le traitement est identique, on crée alors un modèle en utilisant [DAV]...[/DAV], et le nom de l'attribut sera passé en paramètre au modèle :

```
[INC]model.html?NOM=cn[/INC]
[INC]model.html?NOM=givenName[/INC]
```

Le modèle model.html contient :

```
[DAV][ARG]NOM[/ARG][DAV]
```

Il est aussi possible qu'on stocke un nom de l'attribut dans un autre attribut (supposons que l'attribut "attributCourant" contient la valeur "mail", et l'attribut "mail" contient la valeur "wxd@spidernet.tm.fr"), on peut alors récupérer l'adresse de messagerie par :

```
[DAV][AV]attributCourant[/AV][DAV]
```

ce modèle donne

```
[DAV]mail[/DAV]
```

il est équivalent de

```
[AV]mail[/AV]
```

ce qui donne

```
wxd@spidernet.tm.fr
```

On peut aussi utiliser

`[DAV][INC RES]...[/INC]/[DAV]`

pour que le nom de l'attribut soit le résultat d'un autre modèle.

Voir aussi

`[AV]...[/AV]`, `[AVLI]...[/AVLI]`, `[DAVLI]...[DAVLI]`

[DAVLI]...[LI]...[/DAVLI]

Syntaxe

[DAVLI mod1 mod2 ...]nom_de_l'attribut[LI]...[V]...[/DAVLI]

Description

Cette balise est une variante de la balise [AVLI]...[/AVLI].

[DAVLI]...[/DAVLI] est à [AVLI]...[/AVLI] ce que [DAV]...[/DAV] est à [AV]...[/AV].

La différence entre [DAVLI]...[/DAVLI] et [AVLI]...[/AVLI] est que le nom de l'attribut dans le premier peut être dynamique, alors qu'il est "en dur" dans le deuxième. Dans le premier, le nom de l'attribut se spécifie entre [DAVLI] et [LI], alors que dans le deuxième, le nom de l'attribut est spécifié dans [AV name=attribut].

Les exemples fournis dans la description de [DAV]...[/DAV] sont tout aussi valables pour [DAVLI]...[/DAVLI].

[DAVLI][ARG]...[/ARG][LI]...[V]...[/DAVLI]

ou

[DAVLI][AV]attributCourant[/AV][LI]...[V]...[/DAVLI]

ou

[DAVLI][INC RES]...[/INC][LI]...[V]...[/DAVLI]

Voir aussi

[AV]...[/AV], [AVLI]...[/AVLI], [DAV]...[/DAV]

[DECODE]...[/ DECODE]

Syntaxe

[DECODE]...[/DECODE]

Paramètres

Néant

Description

La balise [DECODE]...[/DECODE] permet de décoder son contenu, généralement un URL. Le contenu peut avoir été encodé par une FORM HTML, ou bien par [ENCODE]...[/ENCODE].

Voir aussi:

[ENCODE]...[/ENCODE]

[DELETE]

Syntaxe

[DELETE]

Description

La balise [DELETE] permet de supprimer une entrée dans l'annuaire.

Il faut passer au modèle qui contient la balise [DELETE] le DN de l'entrée à effacer.

Exemples

model.html

```
[DELETE]
```

Pour effacer une entrée il faut appeler le modèle model.html avec le DN en paramètre

```
http://host/model.html?DN=cn=john Smith, o=calice group, c=fr
```

[DN]

Syntaxe

[DN *mod1 mod2 ...*]

Paramètres

mod peut prendre les valeurs suivantes:

RDN= <i>n</i>	Si <i>n</i> est positif, retourne le <i>n</i> ^{ième} RDN du DN, <i>n</i> = 1 désignant le 1 ^{er} RDN (le plus bas dans le DIT, voir la notation UFN). Si <i>n</i> est négatif, retourne le <i>n</i> ^{ième} RDN à partir de la racine.
UP= <i>n</i>	retourner le DN sans les premiers <i>n</i> RDNs, si <i>n</i> = 0, tous les RDNs sont retournés.
UF	retourner la forme "user friendly" d'un DN ou RDN
LEN= <i>n</i>	retourne la partie du DN ayant <i>n</i> RDN à partir de la racine. Si <i>n</i> dépasse le nombre de RDN dans le DN, le DN entier est retourné.
NB_RDN	retourne le nombre de RDN si la valeur est un DN.
BEFORE= <i>pattern</i>	retourne la partie avant <i>pattern</i> . Ex : si la valeur est Service& Département, le modificateur before="&" permet d'obtenir Service.
AFTER= <i>pattern</i>	retourne la partie après <i>pattern</i> . Avec le même exemple que pour before, after="&" donne Département
REPLACE="str1 str2"	Remplacer les «str1» par la chaîne «str2». Si les deux chaînes elles-mêmes contiennent des espaces, les espaces peuvent être exprimées par "\s", un retour chariot par un "\n", un linefeed par un "\r".
RE_REPLACE="/str1/str2/"	Remplacer les «str1» par la chaîne «str2». A la différence de REPLACE, «str1» peut être une expression régulière. On peut utiliser un autre caractère que «/» pour

séparer « str1 » et « str2 » (ex : #str1#str2#). Ceci est utile si « str1 » ou « str2 » contiennent elles-mêmes des « / ».

ASCII	converti les caractères (sans accent) dont le code ASCII est supérieur a 128 en leur équivalent dans un code ASCII inférieur a 128.
ISO8859	converti les caractères T61 en ISO 8859, avec accent.
T61	converti les caractères ISO 8859 (avec accent) en T61.

Si le DN est : cn=dupont, o=Calice, c=fr

```
[DN RDN="1" ] donne cn=Dupont
[DN RDN="2" ] donne o=Calice
[DN RDN="3" ] donne c=fr

[DN RDN="-1" ] donne c=fr
[DN RDN="-2" ] donne o=Calice
[DN RDN="-3" ] donne cn=Dupont

[DN LEN="1" ] donne c=fr
[DN LEN="2" ] donne o=Calice, c=fr
[DN LEN="3" ] donne cn=Dupont, o=Calice, c=fr
[DN LEN="4" ] donne cn=Dupont, o=Calice, c=fr
```

[DEREF]...[/DEREF]

Syntaxe

[DEREF]valeur[/DEREF].

ou *valeur* peut être :

n ⇨ never
a ⇨ always
s ⇨ searching
f ⇨ finding

Description

Spécification de déréférencement des alias pour un modèle.

Dans le modèle :

```
[BASE]
...
[/BASE]
[FILTER]...[/FILTER] (s'ils
existent)
[DEREF]valeur[/DEREF]
```

Dans les paramètres de l'URL

```
model.html?DEREF=valeur
```

Les deux modes n'affectent que le modèle concerné, pas les sous-modèles inclus.

Ordre de priorité (dans l'ordre décroissant) :

- 1) argument : DEREF=*valeur*
- 2) le modèle : [DEREF]*valeur*[/DEREF]

3) le fichier de configuration général : DEREF *valeur*

Remarque : Si nul part le mode n'est spécifié, NEVER est la valeur par défaut.

[EMDL]...[/EMDL]

Syntaxe

[EMDL *DESC*]dn[/EMDL]

Paramètres

DESC utilisable uniquement dans [EMDL *DESC*]dn[/EMDL], il retourne la description d'un modèle de document correspondant à l'entrée dont le DN est spécifié.

Description

Retourne le nom du modèle correspondant à un DN.

La correspondance est spécifiée dans un fichier de configuration. Voir la variable `ENTRY_ATTR_MODEL_FILE` dans le fichier de configuration de WXD.

[ELI]...[/ELI]

Syntaxe

[ELI FLT=filtre1 FLT=filtre2 ...]...[/ELI]

Paramètres

FLT=filtre c'est un filtrage à appliquer à la liste des entrées trouvées dans LDAP/X.500 afin d'interpréter le champ [ELI]...[/ELI] uniquement sur les entrées qui passent le filtre. Le filtre est de la forme "nom_attribut[!]=valeur" où valeur est une expression régulière. La valeur null a une signification spéciale, par exemple, audio=null signifie que l'attribut audio n'est pas défini ou qu'il n'est pas renseigné pour l'entrée en question. Ce modificateur peut apparaître plusieurs fois, chaque filtre étant appliqué dans l'ordre, seules les entrées qui passent tous les filtres sont retournées.

Description

Donne la liste des entrées.

Liste avec en-tête et pied :

```
[ELI]
[HEADER]...[/HEADER]
...
[FOOTER]...[/FOOTER]
[/ELI]
```

Syntaxe similaire pour les autres listes ([ALI], [AVLI]).

Les entêtes et pieds de listes ne seront affichés qu'une seule fois au début et à la fin de la liste respectivement.

Si la liste est vide, rien ne sera affiché.

[ENC]...[/ENC]

Syntaxe

[ENC mod]... [/ENC]

Paramètres

ALL encoder tous les caractères spéciaux.

Description

La balise [ENCODE]...[/ENCODE] permet d'encoder son contenu. Les caractères normaux sont restitués tels quels, les caractères spéciaux sont encodés avec un % suivi de leur code en hexadécimal.

Un espace est encodé en "+".

Cet encodage est nécessaire sur tous les URL :

```
model.html?[ENC]DN=[DN]&ARG=[AV]tel[/AV][ /ENC]
```

Encoder un url (<http://hostport/path?tag=val&tag=val>) consiste à encoder chaque tag et val, par conséquent, si il y a des "=" ou "&" à l'intérieur des tag ou val, ils sont encodés.

On utilise [enc all]...[/enc] pour encoder chaque tag et val.

Mais cette expression est souvent très longue :

```
http://hostport/path?[enc all]tag[/enc]=[enc all]val[/enc]&[enc all]tag[/enc]=[enc all]val[/enc]...
```

Pour simplifier, on utilise [ENC]...[/ENC] qui sépare d'abord "tag=val&tag=val" en couples "tag=val" avant d'appliquer l'encodage :

```
http://hostport/path?[enc]tag=val&tag=val[/enc]
```

Mais ceci fonctionne seulement si aucun tag ni val contienne "&" qui est utilisé pour séparer les couples "tag=val".

Ensuite, on sépare tag et val par le symbole "=", il ne faut donc pas que tag contienne lui-même un "=". Par contre, val peut contenir des "=", car on considère que tous les caractères qui suivent le premier "=" font partie de la valeur, ils sont donc tous encodés.

Règle d'utilisation :

Si un tag ou un val contient un "&" ou un "=", utiliser [ENC ALL]...[/ENC] pour chaque tag et val individuellement :

```
http://hostport/path?[enc all]tag[/enc]=[enc all]val[/enc]&[enc all]tag[/enc]=[enc all]val[/enc]...
```

Sinon, utilise un seul [ENC]...[/ENC] pour l'ensemble des paramètres :

```
http://hostport/path?[enc]tag=val&tag=val[/enc]
```

On peut aussi mixer les deux :

```
model.html?DN=[ENC ALL][DN][[/ENC]&[ENC]arg1=toto&arg2=titi[/ENC].
```

Voir aussi

[DECODE]...[/DECODE]

[ENV]...[/ENV]

Syntaxe

[ENV]*variable d'environnement*[/ENV]

Description

La balise [ENV]...[/ENV] permet de récupérer directement dans un modèle les variables d'environnement du serveur.

Exemple

Si la variable HOME est égale à /usr/local :

```
[ ENV ]HOME [ /ENV ]
```

donne

```
/usr/local
```

On peut mettre d'autres balises entre [ENV]...[/ENV] :

Si l'attribut *var* contient HOME

```
[ ENV ][ AV ]var [ /AV ] [ /ENV ]
```

donne

```
/usr/local
```

[FOOTER]...[/FOOTER]

Syntaxe

[ELI]

[FOOTER]...[/FOOTER]

..

[/ELI]

ou

[AVLI]

[FOOTER]...[/FOOTER]

..

[/ALVI]

ou

[ALI]

[FOOTER]...[/FOOTER]

..

[/ALI]

Description

Affiche un "pied de page" à la liste, celle-ci ne sera affichée qu'une seule fois. Si la liste est vide, le "pied de page" ne sera pas affichée.

Exemple

```
...  
[ELI]  
- [AV]ou[/AV]<BR>  
[FOOTER]sont des services de Calice[/FOOTER]  
[/ELI]  
...
```

S'il existe des entrées on obtient :

- Service Comptabilité
- Service Informatique
- Service Traduction

sont des services de calice.

[GETBEFORE]...[/ GETBEFORE]

Syntaxe

[GETBEFORE PAT="sous-chaîne"]chaîne[/GETBEFORE]

Paramètres

PAT= " sous-chaîne " sous-chaîne de caractères servant de "*séparateur*"

Description

Retourne la chaîne de caractères qui se trouve avant la sous-chaîne de caractères spécifiée dans le paramètre « PAT ».

Exemples

[GETBEFORE PAT="34"] "123456" [/GETBEFORE]

retourne

12

[GETAFTER]...[/GETAFTER]

Syntaxe

[GETAFTER PAT="sous-chaîne"]chaîne[/GETAFTER]

Paramètres

PAT="sous-chaîne" sous-chaîne de caractères servant de "*séparateur*"

Description

Retourne la chaîne de caractères qui se trouve après la sous-chaîne de caractères spécifiée dans le paramètre « PAT ».

Exemples

```
[GETAFTER PAT="34" ] "123456" [/GETAFTER]
```

retourne

```
56
```

[GET_MVAR *paramètre*]...[/GET_MVAR]

Syntaxe

[GET_MVAR]*variable locale*[/GET_MVAR]

[GET_MVAR PVAR]*variable locale*[/GET_MVAR]

[GV] ... [/GV]

Paramètres

PVAR permet de lire une variable du modèle parent, si le modèle courant est inclus dans le modèle père par un [INC RES]...[/INC] ou un [INLINE]...[/INLINE].
Si la variable n'est pas défini, le tag n'a aucun effet.
Un modèle est considéré comme parent s'il inclut le modèle fils avec [INC RES]...[/INC] ou [INLINE]...[/INLINE].
Un modèle qui inclut un autre avec une inclusion simple [INC]...[/INC] n'est pas un modèle parent, car le modèle inclus est considéré comme faisant partie du modèle incluant (comme si le contenu du modèle était directement dans le modèle incluant).
Une variable est accessible par un modèle descendant quelque soit son niveau d'inclusion, à condition qu'il n'y a pas un modèle intermédiaire qui a défini une variable du même nom.

DEC=*valeur* Spécifie le nombre de décimales si la variable est numérique

Description

Renvoie la valeur de la variable locale, spécifiée entre les balises.

Exemple

```
...
[SET_MVAR]Prenom=stephan[/SET_MVAR]
...
<B>Prénom:</B>[GET_MVAR]Prenom[/SET_MVAR]<BR>
...
```

affiche:

```
Prénom:stephan
```

L'exemple suivante illustre la portée des variables locales :

Modèle 1 :

```
[SET_MVAR]V1=1[/SET_MVAR]
[GET_MVAR]V1[/GET_MVAR]: ce qui donne 1

[SET_MVAR]V2=2[/SET_MVAR]
[GET_MVAR]V2[/GET_MVAR]: ce qui donne 2

[SET_MVAR]V3=3[/SET_MVAR]
[GET_MVAR]V3[/GET_MVAR]: ce qui donne 3

[INC RES]modele2[/INC]
```

Modèle 2 :

```
[SET_MVAR]V1=3[/SET_MVAR]
[GET_MVAR]V1[/GET_MVAR]: V1 est local au modele2, ce qui
donne 3

[SET_MVAR PVAR]V2=5[/SET_MVAR] : avec PVAR, on donne 5 a la
variable V2 du modele1
[GET_MVAR]V2[/GET_MVAR] : V2 n'est pas défini dans modele2,
on obtient une chaîne vide

[GET_MVAR PVAR]V2[/GET_MVAR] : on accède a la variable V2 du
modele1, ce qui donne 5

[INC RES]modele3[/INC]
```

Modèle 3:

```
[SET_MVAR PVAR]V1=3[/SET_MVAR]
[GET_MVAR PVAR]V1[/GET_MVAR] : V1 du modele2 qui a masqué V1
du modele1

[SET_MVAR PVAR]V2=3[/SET_MVAR]
[GET_MVAR PVAR]V2[/GET_MVAR] : V2 du modele1

[SET_MVAR PVAR]V3=3[/SET_MVAR]
[GET_MVAR PVAR]V3[/GET_MVAR] : V3 du modele1
```

Valeurs numériques

```
[SET_MVAR]a=1[/SET_MVAR]
[GET_MVAR]a[/GET_MVAR] donne "1",
[GET_MVAR DEC=2]a[/GET_MVAR] donne 1.00
```

Voir aussi

[SET_MVAR]...[/SET_MVAR]

[HEADER]...[/HEADER]

Syntaxe

[ELI]

[HEADER]...[/HEADER]

...

[/ELI]

ou

[AVLI]

[HEADER]...[/HEADER]

..

[/AVLI]

ou

[ALI]

[HEADER]...[/HEADER]

...

[/ALI]

Description

Affiche une entête de liste, celle-ci ne sera affichée qu'une seule fois. Si la liste est vide, l'entête ne sera pas affichée.

Exemple

```
...  
[ELI]  
[HEADER]<B>Liste des services</B>[/HEADER]  
- [AV]ou[/AV]<BR>  
[/ELI]  
...
```

S'il existe des entrées on obtient :

```
Liste des services  
- Service Comptabilité  
- Service Informatique  
- Service Traduction
```

[IF paramètre][THEN][ELSE] [/IF]

Syntaxe

[IF paramètre]...[ELSE]...[/IF]

ou

[IF]conditions[THEN]...[ELSE]...[/IF]

ou (cette dernière sera obsolète)

[IF][COND]conditions[/COND]...[ELSE]...[/IF]

Paramètres

NB_CHLD

NB_CHLD=n

NB_CHLD!=n

NB_CHLD>n

NB_CHLD<n

NB_CHLD>=n

NB_CHLD<=n

Le contenu du champ est pris en compte seulement si le nombre des entrées filles est égal (=), est différent de (!=), est supérieur à (>), ... la valeur n.

FLT

FLT="attribut=valeur", ou

FLT="attribut!=valeur" : le contenu du champ est pris en compte seulement si l'entrée possède l'attribut dont la valeur correspond à valeur (en expression régulière).

null : La valeur null permet de tester la non attribution de valeur à un attribut ou bien l'inexistence d'un attribut.

Description

Le champ [IF]...[THEN]...[ELSE]...[/IF] permet la sélection conditionnelle et dont le contenu ne sera pris en compte que si l'entrée satisfait certaines conditions.

Exemple

Ces modificateurs peuvent apparaître plusieurs fois, auquel cas l'entrée doit satisfaire toutes les conditions.

Utilisation des *nb_chld*

```
...
[if nb_chld>0]
<H2>Groupes:<H2>
[ELI FLT="cn!=^Etat.*"]
<A HREF="groupe.html?[enc]DN=[DN][/enc]">
[AV LG]cn[/AV] ([AV ST aflt="^[0-9]*"cn[/AV])
</A><BR>
[/ELI]
[/if]
...
```

Dans cet exemple, si aucune entrée fille n'a été trouvée, le texte <H2>Groupes:<H2> et la liste des entrées [ELI]...[/ELI] se trouvant à l'intérieur de [if nb_chld>0]...[/if] seront tout simplement ignorés, évitant un affichage du type :

Groupes :

sans rien dans la liste.

Utilisation de *null*:

```
[if flt="jpegPhoto!=null"]
<IMG SRC="photo.mjpeg?DN=[DN]">
[ELSE]
[if flt="jpegPhoto=null"]
Photo non disponible
[/if]
```

Dans cet exemple, si l'attribut jpegPhoto n'est pas défini ou renseigné pour l'entrée, l'inclusion d'images HTML ne se fera pas et c'est le message « Photo non disponible » qui sera retourné. Notons que nous avons utilisé une valeur pré-

définie égale à `null` pour permettre cette sélection. Dans ce contexte, `null` indique aussi bien un attribut non valué qu'un attribut inexistant.

Utilisation de **FLT** :

```
[eli][if flt="cn=^Etat.*"] ... [/if][/eli]
```

Si le modificateur FLT du `[eli]...[/eli]` agit uniquement sur la liste des entrées fille, celui du `[if] ... [/if]` est par contre plus général, et peut s'appliquer aussi bien à l'entrée de base (quand il est à l'extérieur de tous les `[eli]...[/eli]`) qu'aux entrées filles (quand il est à l'intérieur d'un `[eli]...[/eli]`) ; de plus, il peut être placé à n'importe quel endroit dans le modèle.

Les conditions sont du format :

expression	<-- l'expression doit s'évaluer en vrai ou faux
expression & expression	<-- ET logique
expression expression	<-- OU logique
!(expression)	

On peut combiner les expressions comme ci-dessus :
(expression & (expression | !(expression)))

Les expressions peuvent être :

0 : faux
1 : vrai

ou

chiffre opérateur chiffre

où opérateur peut être :

- == : égale
- != : différent de
- : supérieur à
- < : inférieur à
- >= : supérieur et égale à

- <= : inférieur et égale à

chiffre peut être le résultat d'un calcul :

(1 + 2)

(3 - 1)

([AV]salaire[/AV] * 12)

([NB_CHILDREN] / 3)

Pour les chaînes de caractères :

"chaîne" opérateur "chaîne"

où opérateur peut être :

- == : égal
- != : différent de
- : supérieur à
- < : inférieur à
- >= : supérieur et égale à
- <= : inférieur et égale à
- ~ : match

"[AV]telephoneNumber[/AV]"=="26.98.76.89"

("[DN]" ~ ".*InfoPointu\$")

("[AV]cn[/AV]" ~ "\\(\\(M .*\\)\\|\\(Mme .*\\)\\)")

A noter que les "(", ")" et "|" doivent être précédés de deux "\" car un "\" est nécessaire pour la comparaison d'expression elle-même, l'autre est enlevé par l'analyseur syntaxique. La dernière comparaison dans l'exemple est en fait [AV]cn[/AV] avec ((M .*)|(Mme .*)), ce qui signifie que le cn commence par un "M " ou un "Mme ".

Voir aussi

[COND]...[/COND] [ELI]...[/ELI]

`[INC paramètre]...[/INC]`

Syntaxe

`[INC paramètre]fichier[/INC]`

Description

Il y a deux types d'inclusion de modèle, le premier inclut le contenu d'un modèle et le second inclut le résultat final d'un modèle.

Dans le premier cas, il y a une copie simple du contenu du modèle dans le modèle courant, comme s'il était saisi directement dans le modèle courant, dans le second cas, des requêtes préliminaires sont adressées à X.500 et les résultats sont inclus dans la génération du document final avant inclusion.

Le document inclus peut inclure à son tour d'autres documents qui peuvent inclure encore d'autres documents, etc. Ceci permet de rassembler des informations difficiles à obtenir par les requêtes d'un seul modèle. Il faut néanmoins éviter des inclusions récursives qui feront boucler la passerelle à l'infini.

Paramètres

La balise `[INC]référence du document à inclure[/INC]` permet une simple inclusion. La syntaxe de la référence dans `[INC]...[/INC]` est celle de l'URL pour un document sans la partie `http://host/`, c'est-à-dire :

<code>nom_du_modèle?arg1&arg2...</code>

Dans une inclusion simple, les arguments dans l'URL (même présents) seront ignorés ainsi que les paramètres équivalents contenus dans le modèle lui-même (`[BASE]...[/BASE]`, `[FLT]...[/FLT]` ou `[FTAG]...[/FTAG]`, etc.). Seuls ceux du modèle incluant sont pris en compte.

RES l'ajout du modificateur RES permet d'inclure le RESULTAT final du modèle (`[inc RES] ... [/inc]`).
Les arguments (dans l'URL ou dans le modèle) sont utilisées

par le modèle inclus pour générer le résultat. Celui-ci sera inclus dans le modèle père.

TEXT	Dans les deux cas précédents, WXD effectue une analyse syntaxique du modèle inclus. Par conséquent, quand on inclut un fichier texte sans aucune balise de modèle, il est préférable d'utiliser le modificateur TEXT qui est une version plus rapide de l'inclusion mais qui ne garantit plus la récupération des valeurs de tous les attributs dans le modèle à inclure.
ALL_ARGS	Indique que tous les arguments du modèle courant seront passés au modèle inclus. Ce qui permet, lors qu'on utilise des modèles intermédiaires, de transmettre aux modèles suivants tous les arguments initiaux sans les spécifier explicitement un à un par [ARG]...[/ARG]. S'il y a conflit, les arguments spécifiés après le nom du modèle est prioritaire par rapport aux arguments du modèle courant.
EMPTY	Indique que le modèle inclus doit retourner une chaîne vide. Dans certains cas, nous concevons des modèles pour effectuer des opérations plus ou moins complexes. Pour des raisons de lisibilité, nous introduisons des retours à la ligne, des espaces ou des tabulations dans le code. Par conséquent, le contenu retourné par le modèle peut contenir beaucoup d'espaces inutiles. Dans ce cas, nous pouvons utiliser "EMPTY" pour que le modèle ne renvoie rien. Si le modèle doit transmettre des informations au modèle père, on peut le faire par l'intermédiaire des variables ([SV]...[/SV] et [GV]...[/GV]).
DECODE_URL	Indique qu'il faut décoder les paramètres passés après "?". Normalement, les paramètres sont passés en claire au sous modèle, mais dans certains cas où on est obligé d'encoder les paramètres avant de les passer au sous modèle, il faut que le sous modèle les décode avant de les utiliser. Voir l'exemple ci-après.

Exemples

Par exemple, pour inclure le modèle `ingenieur.html`, on peut utiliser :

```
[INC]ingenieur.html[/INC]
```

Ce champ va inclure le contenu du modèle `ingenieur.html` dans le modèle présent. Les paramètres de recherche contenus dans le modèle seront ignorés. La référence au document à inclure peut contenir des champs de valeur qui seront interprétés pour obtenir la référence finale. Ce qui permet d'avoir une référence dynamique en fonction des entrées trouvées dans le document présent. Par exemple, si on veut inclure le document final résultant de l'application du modèle `ingenieur.html` avec un ingénieur trouvé dans le document présent, on peut formuler le champ suivant :

```
[INC RES]ingenieur.html?DN=[DN][/INC]
```

Supposant que `[DN]` donne `cn=John Smith`, on obtient alors:

```
[INC RES]ingenieur.html?DN=cn=john Smith[/INC]
```

Nous pouvons utiliser aussi un attribut dont la valeur est un DN :

```
[INC RES]ingenieur.html?DN=[AV]seeAlso[/AV][/INC]
```

Si un attribut contient une référence complète, il est possible d'utiliser la formulation suivante :

```
[INC][AV]refDoc[/AV][/INC]
```

Pour inclure un fichier texte ordinaire, nous pouvons écrire :

```
[inc TEXT]copyright.html[/inc]
```

Retransmission de tous les arguments aux modèles inclus

Si un modèle reçoit un ensemble d'arguments, envoyés soit dans l'URL, soit par un formulaire HTML, et qu'on souhaite transmettre tous ces arguments à un modèle d'inclusion, on utilise ALL_ARGS.

Supposons qu'on appelle un modèle par :

```
http://host/m1.html?DN=<dn>&ATTRNAME=sn&ATTRVAL=me
```

Le modèle "m1.html" contenant :

```
[INC ALL_ARGS RES]m2.html?arg1=val1&arg2=val2[/INC]
```

Le modèle "m2.html" va recevoir, en plus des deux arguments "arg1" et "arg2", tous les arguments du modèle m1.html, c'est à dire "DN", "ATTRNAME" et "ATTRVAL".

Ceci est équivalent de :

```
[INC RES]
m2.html?arg1=val1&arg2=val2&DN=[DN]&ATTRNAME=[ARG]ATTRNAME[/ARG]&ATTRVAL=[ARG]ATTRVAL[/ARG][ /INC]
```

Mais si un argument contient des données de grand volume ou des données binaires (comme un fichier envoyé par un formulaire HTML), on ne peut plus remplacer comme dans l'exemple précédent ALL_ARGS par une suite de [ARG]...[/ARG].

Encodage/Décodage des paramètres

Les paramètres d'un sous modèle sont séparés par un "&", si un paramètre contient lui-même un "&", il faut l'encoder avec [ENC ALL]...[/ENC] pour qu'il n'introduise pas de confusion dans la séparation des paramètres.

```
[INC RES] model.html?arg1=val1&arg2=[ENC
ALL] (&(cn=durant)(age>20)) [/ENC] [/INC]
```

Dans ce cas, il faut signaler au sous modèle qu'il faut décoder les paramètres avant de les utiliser, on utilise alors le modificateur DECODE_URL:

```
[INC RES DECODE_URL] model.html?arg1=val1&arg2=[ENC
ALL] (&(cn=durant)(age>20)) [/ENC] [/INC]
```

Voir aussi

[INLINE]..[/INLINE], [PARAM]...[/PARAM]

[*INLINE* *modificateurs*][*PARAM*]*params*[/*PARAM*] [*/INLINE*]

Syntaxe

[*INLINE* *modificateurs*][*PARAM*]*params*[/*PARAM*]*contenu*[/*INLINE*]

Paramètres

ALL_ARGS	Indique que tous les arguments du modèle courant seront passés au modèle inclus. Ce qui permet, lors qu'on utilise des modèles intermédiaires, de transmettre aux modèles suivants tous les arguments initiaux sans les spécifier explicitement un à un par <code>[ARG]...[/ARG]</code> . S'il y a conflit, les arguments spécifiés après le nom du modèle est prioritaire par rapport aux arguments du modèle courant.
EMPTY	Indique que le modèle inclus doit retourner une chaîne vide. Dans certains cas, nous concevons des modèles pour effectuer des opérations plus ou moins complexes. Pour des raisons de lisibilité, nous introduisons des retours à la ligne, des espaces ou des tabulations dans le code. Par conséquent, le contenu retourné par le modèle peut contenir beaucoup d'espaces inutiles. Dans ce cas, nous pouvons utiliser "EMPTY" pour que le modèle ne renvoie rien. Si le modèle doit transmettre des informations au modèle père, on peut le faire par l'intermédiaire des variables (<code>[SV]...[/SV]</code> et <code>[GV]...[/GV]</code>).
DECODE_URL	Indique qu'il faut décoder les paramètres passés après "?". Normalement, les paramètres sont passés en claire au sous modèle, mais dans certains cas où on est obligé d'encoder les paramètres avant de les passer au sous modèle, il faut que le sous modèle les décode avant de les utiliser. Voir l'exemple dans <code>[INC]...[/INC]</code> .

params : Correspond aux paramètres que l'on veut passer au modèle local.

contenu : Contenu d'un petit modèle.

Description

Cette balise est ajoutée à la balise [INC]...[/INC] pour éviter de créer des petits fichiers d'inclusion. Grâce à cette balise, on peut écrire le contenu d'un petit modèle directement dans le modèle père. Les [INLINE] peuvent être imbriqués.

Exemples

Si le modèle "petit_modele.html" contient :

```
[READ]
[AV]cn[/AV]
```

alors les balises :

```
[INC RES]petit_modele.html?DN=cn=Durand[/INC]
```

peuvent être remplacées par :

```
[INLINE] [PARAM]DN=cn=Durand[/PARAM]
[READ]
[AV]cn[/AV]
[/INLINE]
```

On met les paramètres passés au modèle (DN=cn=Durand) entre [PARAM]...[/PARAM], et le contenu du fichier petit_modele.html juste après (avant [/INLINE]).

On peut récupérer tous les arguments du modèle courant pour les passer au modèle inline :

```
[INLINE ALL_ARGS] [PARAM]DN=cn=Durand[/PARAM]
[READ]
[AV]cn[/AV]
[/INLINE]
```

Voir aussi

[INC]..[/INC]

[INPUT_ATTR]attr=value[/INPUT_ATTR]

Syntaxe

[INPUT_ATTR]attribut=value[/INPUT_ATTR]

Description

Si pour un modèle donné, on a systématiquement besoin des valeurs d'attributs (ex: création d'une entrée avec l'ensemble des classes), au lieu de les passer en paramètres du modèle par ATTRNAME=attribut&ATTRVAL=value, on les met directement dans le modèle :

```
[INPUT_ATTR]objectClass=top[/INPUT_ATTR]
[INPUT_ATTR] objectClass=person[/INPUT_ATTR]
```

Voir aussi

[MODIFY], [ADD]

[LDAP_HOST]adresse_ip[/LDAP_HOST]

Syntaxe

[LDAP_HOST]adresse_ip[/LDAP_HOST]

Description

Spécifier l'adresse IP du serveur LDAP.

Normalement, l'adresse IP du serveur LDAP est spécifié dans le fichier de configuration de WXD, elle est globale à toutes les requêtes.

Si pour une requête donnée, on souhaite changer de serveur LDAP, on peut spécifier son adresse dans le modèle avec [LDAP_HOST].

Cette adresse ne sera pas prise en compte si le modèle reçoit en paramètre un argument LDAP_HOST :

```
model.html?LDAP_HOST=192.168.1.1
```

Dans ce cas, c'est l'argument du modèle (LDAP_HOST=192.168.1.1) qui a la priorité sur la balise ([LDAP_HOST]...[/LDAP_HOST]).

Cette balise peut être utilisée en combinaison avec [LDAP_PORT].

Si LDAP_PORT n'est pas fourni, le port par défaut (389) est suppose.

Si LDAP_HOST n'est pas fourni, le host spécifié dans le fichier de configuration (wxd.conf) est utilisé.

Si ni LDAP_HOST, ni LDAP_PORT est fourni, c'est le host et le port spécifiés dans le fichier de configuration (wxd.conf) qui sont utilisés.

Voir aussi

[LDAP_PORT]...[LDAP_PORT]

[LDAP_PORT]numero_port[/LDAP_PORT]

Syntaxe

[LDAP_PORT] numero_port [/LDAP_PORT]

Description

Spécifier le numéro de port du serveur LDAP.

Normalement, le numéro de port du serveur LDAP est spécifié dans le fichier de configuration de WXD, il est global à toutes les requêtes.

Si pour une requête donnée, on souhaite changer de port LDAP, on peut spécifier le port dans le modèle avec [LDAP_PORT].

Ce port ne sera pas pris en compte si le modèle reçoit en paramètre un argument LDAP_PORT :

```
model.html?LDAP_PORT=1234
```

Dans ce cas, c'est l'argument du modèle (LDAP_PORT=1234) qui a la priorité sur la balise ([LDAP_PORT]...[/LDAP_PORT]).

Cette balise peut être utilisée en combinaison avec [LDAP_HOST].

Voir aussi

[LDAP_HOST]...[/LDAP_HOST]

[LIST]

Description

Effectue une lecture de l'entrée de type [READ], ainsi qu'une recherche au niveau fille de l'entrée.

[MODIFY]

Syntaxe

[MODIFY]

[INPUT_ATTR]+/- *attribut* = *valeur de l'attribut*[/INPUT_ATTR]

Paramètres

En paramètre d'un modèle :

DN dn de l'entrée à modifier, paramètre obligatoire.

ATTRNAME nom de l'attribut à modifier précédé d'un :

+ pour un ajout

- pour un effacement

rien pour le remplacement

ATTRVAL valeur de l'attribut à modifier

Description

Permet de modifier les valeurs d'un annuaire.

Deux possibilités existent :

Si on sait qu'un modèle sera toujours utilisé pour effectuer des modifications avec des valeurs et des attributs fixes, on utilise les balises [INPUT_ATTR][INPUT_ATTR]

Si on veut une modification dynamique, on peut utiliser un modèle contenant seulement la balise [MODIFY] et appeler celui-ci avec des paramètres qui peuvent être fournis soit dans l'URL, soit dans les champs d'un formulaire HTML et transmis par les méthodes GET ou POST de la commande *submit* de HTML.

Les attributs doivent être fournis comme des couples.

L'ordre des paramètres est important, car on attend la valeur d'un attribut immédiatement après son nom.

```
ATTRNAME=commonName
ATTRVAL=Paul Durant
ATTRNAME=-roomNumber
ATTRVAL=10
ATTRNAME+=roomNumber
ATTRVAL=11
...
```

Exemples

Mise à jour directement de l'annuaire par un modèle contenant *en dur* les modifications à effectuer :

```
[MODIFY]
[BASE]
cn= john Smith,
ou= personnel,
o = calice group,
c=fr
[/BASE]
[INPUT_ATTR]+commonName= Paul Durant[/INPUT_ATTR]
[INPUT_ATTR]-roomNumber=10[/INPUT_ATTR]
```

Mise à jour par un passage de paramètres a un modèle:

Si le modèle ne contient que la balise [MODIFY]

modele.html

```
[MODIFY]
```

on peut effectuer les modifications en appelant le modèle avec les données en paramètres :

```
http://host/model.html?DN=...&ATTRNAME=commonName&ATTRVAL=Paul
Durant
```

soit en utilisant un formulaire :

```
[READ]
<HTML>
..
<FORM method="POST" action="model.html">
  <input type="hidden" name="DN" value="[DN]">
  <input type="hidden" name="ATTRNAME"
value="+telephoneNumber">
  Numéro de Téléphone :
  <input type="text" name="ATTRVAL"><BR>
  <input type="submit" value="OK">
</FORM>
</HTML>
```

Ce formulaire va ajouter un numéro de téléphone au champ *telephoneNumber* , si on veut effacer le champ, on utilise le signe - au lieu du +.

Exemple de chaînage des modifications

Supposons que dans une entrée "person", on a l'attribut seeAlso qui pointe sur la fonction de la personne, et que dans l'entrée de la fonction, on a roleOccupant qui pointe sur la personne.

Quand on supprime le champ seeAlso de la personne, il faut donc supprimer le DN de la personne dans le champ roleOccupant de la fonction.

Solution : on crée un modèle qui recupere le seeAlso de la personne pour connaître la fonction, ensuite on inclut un modèle qui supprime le DN de la personne dans le roleOccupant de la fonction, enfin on inclut un autre modèle qui supprime le DN de la fonction dans seeAlso de la personne.

```
http://host/mod_person.html?DN=<dn_personne>
```

mod_person.html:

```
[READ]
[INC RES]
sup_person_from_roleOccupant.html?DN=[AV]seeAlso[/AV]&ATTRNAME=-roleOccupant&ATTRVAL=[DN]
[/INC]
[INC RES]
sup_fonc_from_seeAlso.html?DN=[DN]&ATTRNAME=-seeAlso&ATTRVAL=[AV]seeAlso[/AV]
[/INC]
```

sup_person_from_roleOccupant.html

```
[MODIFY]
```

sup_fonc_from_seeAlso.html

```
[MODIFY]
```

A part la commande [MODIFY], chaque modèle peut contenir d'autres informations, mais ici on n'en a pas besoin, ces deux modèles sont donc identiques. On peut en effet créer un modèle type pour chaque type de mise à jour.

Dans l'exemple précédent, on supprime la valeur de « seeAlso ».

Pour remplacer le « seeAlso » par un autre :

```
http://host/replace_seeAlso.html?DN=<dn_person>&new_seeAlso=<new_dn_fonction>
```

replace_seeAlso.html :

```

[READ]

[REM]
Commentaire : On récupère l'ancien « seeAlso »
avec [READ], on supprime le DN de la personne
dans roleOccupant de la fonction (comme dans
l'exemple précédent).
[/REM]
[INC RES]modify.html?DN=[AV]seeAlso[/AV]&
ATTRNAME=roleOccupant&ATTRVAL=[DN][/INC]

[REM]
On rajoute le DN de la personne dans le
roleOccupant de la nouvelle fonction (fourni en
paramètre dans l'URL d'origine)
[/REM]
[INC RES]modify.html?DN=[ARG]new_seeAlso[/ARG]&
ATTRNAME=roleOccupant&ATTRVAL=[DN]
[/INC]

[REM]
On supprime le DN de l'ancienne fonction dans le
« seeAlso » de la personne et on rajoute le DN
de la nouvelle fonction
[/REM]
[INC RES]modify.html?DN=[DN]&ATTRNAME=-
seeAlso&ATTRVAL=[AV]seeAlso[/AV]&ATTRNAME=seeAls
o&ATTRVAL=[ARG]new_seeAlso[/ARG][/INC]

```

Pour ne pas avoir à créer un modèle « modify.html » qui ne contient que [MODIFY], on peut utiliser les modèles inline, c'est-à-dire au lieu de faire :

```

[INC RES]modify.html?DN=[AV]seeAlso[/AV]&ATTRNAME=
roleOccupant&ATTRVAL=[DN][/INC]

```

On fait :

```
[ INLINE ]  
[ PARAM ] DN=[ AV ] seeAlso [ / AV ] &ATTRNAME=-  
roleOccupant&ATTRVAL=[ DN ] [ / PARAM ]  
[ MODIFY ]  
[ / INLINE ]
```

[MODIFYRDN]

Syntaxe

[MODIFYRDN]

Paramètres

NEWRDN= nouveau DN :

nouveau DN de l'entrée que l'on veut modifier, ce paramètre est passé à un modèle en plus du DN actuel de l'entrée à modifier.

Description

Permet de modifier le RDN d'une feuille de l'arbre LDAP/X500. Il faut fournir le DN et le nouveau RDN en argument à un modèle.

Exemples

Le modèle qui est appelé pour effectuer la modification doit avoir la forme suivante :

model.html :

```
[MODIFYRDN]
```

On effectue la modification :

- soit en passant directement les paramètres à un modèle:

```
http://host/model.html?DN=cn=john Smith, o=calice group, c=fr&NEWRDN=cn=johnny Smith
```

- soit en utilisant un formulaire :

```
[READ]
<HTML>
..
<FORM method="POST" action="model.html">
  <input type="hidden" name= "DN" value="[DN]">
  <input type="hidden" name= "NEWRDN" value="cn=johnny
Smith">
<input type="submit" value="OK">
</FORM>
</HTML>
```

[NB_CHILDREN]

Syntaxe

[NB_CHILDREN]

Description

A l'extérieur des balises [ELI]...[/ELI], [NB_CHILDREN] donne le nombre des entrées filles de l'entrée de base, à l'intérieur d'un [ELI]...[/ELI], il donne le nombre des entrées filles de l'entrée courante.

[ON_ERROR].. [/ON_ERROR]

Syntaxe

[ON_ERROR *erreur erreur ...*]message[/ON_ERROR]

Paramètres

La liste des erreurs possibles est la suivante :

LDAP_NOT_FOUND	LDAP_INVALID_CREDENTIALS
LDAP_OPEN_FAILURE	LDAP_INSUFFICIENT_ACCESS
NO_DN	LDAP_BUSY
NO_ATTRS	LDAP_UNAVAILABLE
LDAP_OPERATIONS_ERROR	LDAP_UNWILLING_TO_PERFORM
LDAP_TIMELIMIT_EXCEEDED	LDAP_LOOP_DETECT
LDAP_COMPARE_FALSE	LDAP_NAMING_VIOLATION
LDAP_COMPARE_TRUE	LDAP_OBJECT_CLASS_VIOLATION
LDAP_STRONG_AUTH_NOT_SUPPORTED	LDAP_NOT_ALLOWED_ON_NONLEAF
LDAP_STRONG_AUTH_REQUIRED	LDAP_NOT_ALLOWED_ON_RDN
LDAP_PARTIAL_RESULTS	LDAP_ALREADY_EXISTS
LDAP_NO_SUCH_ATTRIBUTE	LDAP_NO_OBJECT_CLASS_MODS
LDAP_UNDEFINED_TYPE	LDAP_RESULTS_TOO_LARGE
LDAP_INAPPROPRIATE_MATCHING	LDAP_OTHER
LDAP_CONSTRAINT_VIOLATION	LDAP_SERVER_DOWN
LDAP_TYPE_OR_VALUE_EXISTS	LDAP_LOCAL_ERROR
LDAP_INVALID_SYNTAX	LDAP_ENCODING_ERROR
LDAP_NO_SUCH_OBJECT	LDAP_DECODING_ERROR
LDAP_ALIAS_PROBLEM	LDAP_TIMEOUT
LDAP_INVALID_DN_SYNTAX	LDAP_AUTH_UNKNOWN
LDAP_IS_LEAF	LDAP_FILTER_ERROR
LDAP_ALIAS_DEREF_PROBLEM	LDAP_USER_CANCELLED
NAME_ERROR(n)	LDAP_PARAM_ERROR
LDAP_INAPPROPRIATE_AUTH	LDAP_NO_MEMORY

Description

Traitement d'erreur dans les modèles

Cette balise indique au modèle s'il rencontre l'"*erreur*", au lieu de retourner le message d'erreur par défaut, retourne "message" fourni.

Si "*erreur*" n'est pas indiquée, "*message*" est affiché quand on rencontre n'importe quelle erreur.

Exemples

```
[READ]
[ON_ERROR "LDAP_TIMEOUT"]Temps limit passe[/ON_ERROR]
...
```

Si un timeout est rencontré, "Temps limit passe" sera retourné au client HTTP.

```
[LIST]
[ON_ERROR]Une erreur est rencontrée[/ON_ERROR]
```

Le message "Une erreur est rencontrée" quand n'importe quelle erreur s'est produite dans la requête.

On peut avoir une liste de ON_ERROR :

```
[ON_ERROR erreur1]message1[/ON_ERROR]
[ON_ERROR erreur2 erreur3]message2[/ON_ERROR]
```

On peut inclure d'autres balises dans le message :

```
[LIST]
[ON_ERROR]Une erreur est rencontrée sur [DN], l'argument
passé est [ARG]NB[/ARG][[/ON_ERROR]
```

[READ]

Description

Effectuer une lecture de l'entrée en récupérant les valeurs des attributs se trouvant dans le modèle.

[REM]...[/REM]

Syntaxe

[REM]commentaires[/REM]

Description

Champ commentaire. Tout ce qui est inclus entre les deux balises est ignoré.

Exemple

```
[REM] **** Boucle ***** [/REM]
```

[REMOVE]...[/REMOVE]

Syntaxe

[REMOVE]path[/ REMOVE]

Description

Supprimer un fichier sur le serveur.

Exemple

```
[REM]/tmp/temp_file.html[/REM]  
[REM][AV]doc_path[/AV][ /REM]
```

[RETURN_ATTRS]...[/RETURN_ATTRS]

Syntaxe

[RETURN_ATTRS]attr1,attr2[/RETURN_ATTRS]

Description

Ce tag est introduit pour une optimisation, son utilisation n'est donc pas obligatoire.

Normalement, WXD collecte automatiquement les attributs à récupérer dans les tags [AV]...[/AV] ou [AVLI]...[/AVLI]. C'est seulement les valeurs de ces attributs qui sont récupérées de l'annuaire.

Mais quand un modèle inclus un autre avec [INC]...[/INC] (notons que le problème ne se pose pas pour [INC RES]...[/INC] ou [INC TEXT]...[/INC]), WXD ne connaît pas les attributs qui sont inclus dans le sous-modèle, il demande alors à l'annuaire de renvoyer TOUS les attributs.

Si une requête renvoie un nombre important d'entrées, le volume de données ainsi récupérées peut être très important (nombre d'entrées x tous les attributs).

Pour limiter les attributs au strict nécessaire, nous utilisons [RETURN_ATTRS]attr1,attr2[/RETURN_ATTRS], dans ce cas, les attributs ne sont plus collectés automatiquement, WXD ne récupère que les attributs listés dans ce tag.

Ce tag est à placer avec les tags de requête [BASE]...[/BASE], [FILTER]...[/FILTER], etc.

En absence de ce tag, tout fonctionne comme avant.

Ce tag influence uniquement les entrées filles récupérées par [LIST] ou [SEARCH] (celles affichées dans [ELI]...[/ELI]), il n'affecte pas l'entrée de base (les attributs de l'entrée de base continuent d'être collectés dans les [AV]...[/AV], et si un [INC]...[/INC] est utilisé, tous les attributs de l'entrée de base sont récupérés, qu'il y ait la présence de [RETURN_ATTRS] ou non. Du fait qu'il y a toujours une seule entrée de base pour une requête, ce n'est pas pénalisant).

[SAVE][PATH]path[/PATH]contenu[/SAVE]

Syntaxe

[SAVE][PATH]path[/PATH]contenu[/SAVE]

Sauvegarde le contenu dans un fichier path (sur le serveur)

[REMOVE]path[/REMOVE]

Supprime un fichier dont le chemin est path (sur le serveur)

path et contenu peuvent être obtenus d'autres tags.

Description

Sauvegarde et suppression des fichiers

Exemple

```
[SAVE][PATH]c:\tmp\photo.jpg[/PATH][AV
DUMP]jpegPhoto[/AV][SAVE]
[REMOVE][AV]fiche[/AV][REMOVE]
```

[SEARCH]

Description

Effectue une lecture de l'entrée de type [READ], ainsi qu'une recherche dans le sous-arbre de l'entrée.

[SENDMAIL]

Syntaxe

```
[SENDMAIL][TO]...[FROM]...[SUBJECT]...[MSG]...[MSGFILE]...[ATTACH]...[/ATTACH][ATTACHFILE]...[/ATTACHFILE][SENDMAIL]
```

Description

Envoie d'un e-mail par WXD.

[TO] peut contenir une liste d'adresse séparées par des "," (ex: abc@spidernet.tm.fr,def@spidernet.tm.fr).

[MSG] ou [MSGFILE] est le corps du message.

[MSG] et [MSGFILE] sont exclusifs.

[ATTACH]...[/ATTACH] et [ATTACHFILE]...[/ATTACHFILE] sont optionnels et peuvent se répéter pour attacher plusieurs fichiers.

[MSG] attend que ce qui le suit soit le texte du message lui-même, alors que [MSGFILE] attend un nom du fichier.

[ATTACH]data[/ATTACH] : data est le contenu du fichier à attacher entant que fichier. En absence du nom du fichier, WXD générera un nom aléatoire.

[ATTACHFILE]nom_du_fichier[/ATTACHFILE] : nom_du_fichier est le chemin absolu du fichier (sur le serveur) à attacher.

Cas spécial pour les fichiers envoyés par une FORM HTML (ex: <input type=file name=photo>) :

Si [ATTACH]...[/ATTACH] contient un seul [ARG]...[/ARG], et qu'il correspond à un fichier envoyé par le Web, le nom du fichier utilisé est celui du fichier d'origine.

On peut combiner [SAVE]...[/SAVE] pour stocker des données dans un fichier avant de l'envoyer en fichier attaché :

```
[ATTACHFILE]c:\tmp\photo.jpg[/ATTACHFILE]
```

En combinant avec [AV]...[/AV], ou bien [INC RES]model[/INC], on peut aussi envoyer le contenu des attributs ou le résultat d'un modèle soit comme le corps du message, soit comme fichier attaché.

Exemples

1) FORM HTML

```
<html>
<body>
<form action="sendmail.html" enctype="multipart/form-data"
method=post>
<input type=hidden name=DN value="cn=Durant,
o=spidernet.tm.fr, c=fr">
Subject : <input type="text" name="subject" value=""><br>
Message : <input type=textarea name="message"><br>
File : <input name="file1" type="file"><br>
File : <input name="file2" type="file"><br>
<input type="submit" value="Send">
</form>
</body>
</html>
```

2) sendmail.html

```
[READ]
[SENDMAIL]
[TO][AV]email[/AV]
[FROM]z.wang@local
[SUBJECT][ARG]subject[/ARG]
[MSG][ARG]message[/ARG]
[ATTACH][ARG bin]file1[/ARG][[/ATTACH]
[ATTACH][ARG bin]file2[/ARG][[/ATTACH]
[/SENDMAIL]
```

En récupérant une liste des adresses email soit par [ELI]...[/ELI], soit par un [INC RES]...[/INC], on peut envoyer un message à un ensemble de personnes :

```
[TO][ELI][AV]email[/AV],[/ELI]
```

[SERVER_TIME]

Temps serveur

Description

[SERVER_TIME] retourne le temps serveur en seconde depuis 01/01/1970.

[SERVER_TIME UTCTIME] retourne le même temps mais sous format UTC : `yyyymmddhhmmss`

[SET_MVAR *paramètre*]...[/SET_MVAR]

Syntaxe

[SET_MVAR]*variable locale*=*valeur*[/SET_MVAR]

ou

[SV] ... [/SV]

Paramètre

PVAR	permet d'affecter une variable du modèle parent si le modèle courant est inclus dans le modèle père par un [INC][INC] ou un [INLINE][INLINE]. Si la variable n'est pas définie, le TAG n'a aucun effet.
ARRAY	tableau des valeurs
SORT	tri d'un tableau.
REVERSE	inverser le tri.

Un modèle est considéré comme parent s'il inclut le modèle fils avec [INC RES]...[/INC] ou [INLINE]...[/INLINE].

Un modèle qui inclut un autre avec une inclusion simple [INC]...[/INC] n'est pas un modèle parent, car le modèle inclus est considéré comme faisant partie du modèle incluant (comme si le contenu du modèle était directement dans le modèle incluant).

Une variable est accessible par un modèle descendant quelque soit son niveau d'inclusion, à condition qu'il n'y a pas un modèle intermédiaire qui a défini une variable du même nom.

Description

Affecte une valeur à une variable locale. Les chaînes de caractères doivent être définies entre "". La valeur peut être une expression numérique.

Exemple

```
...  
[SET_MVAR]Prenom=stephan[/SET_MVAR]  
...  
<B>Prénom:</B>[GET_MVAR]Prenom[/SET_MVAR]<BR>  
...
```

affiche :

```
Prénom:stephan
```

Expressions numériques:

```
[SET_MAR]a=1+2[/SET_MAR] : a est affecté la valeur 3.0000.  
[SET_MAR]a="1+2"[/SET_MAR] : a est affecté la chaîne 1+2.  
[SET_MAR]b=(19-3)/(4*9+(10-1))[/SET_MVAR]
```

Exemples des tableaux :

Tableau des chaînes de caractères

```
[SV ARRAY]tab1="a,b,c,d"[/SV]
```

Tableau des chaînes de caractères avec ":" comme séparateur, le séparateur par défaut est le ",".

```
[SV ARRAY SEP=":" ]tab2="a:b:c:d"[/SV]
```

Tableau des chiffres

```
[SV ARRAY SEP=":" ]tab3=1:2:3:4[/SV]
```

Le nombre d'éléments dans le tableau

```
[GV ARRAY SIZE]tab1[/GV]
```

Le premier élément du tableau

```
[GV ARRAY]tab1(1)[/GV]
```

Entre [SV]...[/SV] et [GV]...[/GV], on peut avoir d'autres tags:

```
[SV ARRAY]tab=1,(2+3),[GV]foo[/GV][[/SV] : si foo = 3,  
[GV ARRAY]tab(1)[/GV] donne 1,  
[GV ARRAY]tab(2)[/GV] donne 5,  
[GV ARRAY]tab(3)[/GV] donne 3.
```

Le code suivant affiche les éléments d'un tableau sur chaque ligne:

```
[SV ARRAY]array=2,3,4,(5+2),6[/SV]  
[SV]len=[GV ARRAY SIZE]array[/GV][[/SV]  
[SV]i=1[/SV]  
[WHILE]([GV]i[/GV]<=[GV]len[/GV])[DO]  
[GV ARRAY]array([GV]i[/GV])[/GV]<br>  
[SV]i=[GV]i[/GV]+1[/SV]  
[/WHILE]
```

Tri d'un tableau

```
[SV ARRAY SORT]tab="t,h,s"[/SV] donne h,s,t  
[SV ARRAY SORT REVERSE]tab="t,h,s"[/SV] donne t,s,h  
  
[SV ARRAY SORT]tab=2,5,17,9[/SV] donne 17,2,5,9  
[SV ARRAY DEC=2 SORT]tab=2,5,17,9[/SV] donne 02,05,09,17  
[SV ARRAY DEC=2 SORT REVERSE]tab=2,5,17,9[/SV] donne  
17,09,05,02
```

On peut utiliser DEC dans [SV DEC=3]...[/SV] pour compléter un chiffre par 0 si le nombre de chiffres est inférieur à 3 (pour l'exemple).

On ne peut utiliser que des entiers dans [SV]

Voir aussi

```
[GET_MVAR][[/GET_MVAR]
```

[SIZELIMIT] n[/SIZELIMIT]

Syntaxe

[SIZELIMIT]n[/SIZELIMIT]

Description

Paramètre de requête

Cette balise spécifie le nombre d'entrées maximum à retourner par une recherche [LIST] ou [SEARCH].

Comme pour les autres paramètres, si le modèle reçoit en argument SIZELIMIT=m, par exemple :

model.html?SIZELIMIT=10

La valeur de l'argument (10) emporte sur la valeur (n) spécifiée dans la balise [SIZELIMIT]n[/SIZELIMIT] à l'intérieur du modèle, si elle est présente.

En absence à la fois de l'argument et de la balise, c'est la variable SIZELIMIT dans le fichier de configuration qui détermine la limite. Cette dernière est globale à tous les modèles.

Si null part SIZELIMIT n'est spécifié, aucune limitation n'est imposée par WXD.

Dans tous les cas, si le serveur LDAP impose de son côté une limite, WXD recevra au plus autant de résultats que cette limite.

Voir aussi

[TIMELIMIT]n[/TIMELIMIT]

[SIZE_LIMIT_EXCEEDED]

Syntaxe

[IF][COND][SIZE_LIMIT_EXCEEDED][/COND][/IF]

Description

Booléen , renvoie 1 si la taille maximum est atteinte lors d'une requête.

[SORTBY]...[/SORTBY]

Syntaxe

[SORTBY paramètre]...[/SORTBY]

Paramètres

HIDDEN : indique que le champ de tri ne doit pas s'afficher.

REVERSE : inverse l'ordre de tri.

Description

Le tri se fait sur le champ [SORTBY]...[/SORTBY] à l'intérieur des balises [AVLI] [/AVLI] et [ELI]...[/ELI].

Exemple

Tri sur le champ NNI :

```
[ELI]
...
[SORTBY][AV]NNI[/AV][ /SORTBY]
...
[/ELI]

Tri sur NNI puis roomNumber :
[ELI]
...
[SORTBY REVERSE][AV]NNI[/AV][AV]roomNumber[/AV][ /SORTBY]
...
[/ELI]
```

Tri sur le RDN de seeAlso, cacher le RDN dans la sortie de [ELI]...[/ELI]

```
[ELI]
...
[SORTBY HIDDEN][AV RDN=1]seeAlso[AV][/SORTBY]
...
[/ELI]
```

Tri sur valeur numérique :

```
[ELI]
...
[SORTBY][AV FORMAT="%.4d"]salaire[AV][/SORTBY]
...
[/ELI]
```

Si "1", "11" et "2" sont triés tels quels, on obtient :

```
1
11
2
```

Pour résoudre le problème, on donne un format "%.4d", chaque nombre est passé en 4 chiffres, le tri donne :

```
0001
0002
0011
```

On peut aussi trier sur des attributs des entrées dont on a seulement le DN, par exemple, si on a une liste de fonctions, chaque fonction possède un attribut "roleOccupant" qui pointe sur des personnes. On peut alors trier la liste de fonction sur (par exemple) l'âge des personnes :

```
[ELI]
...
[SORTBY][INC
RES]age.html?DN=[AV]roleOccupant[/AV][/INC][/SORTBY]
...
[ELI]
```

Dans le fichier "age.html", on inclut uniquement

```
[READ]
[BASE]...[/BASE]
[AV]age[/AV]
```

Avec ce fichier,

[**INC** **RES**]age.html?DN=[AV]roleOccupant[/AV][/**INC**] retourne l'âge de la personne, qui sera utilisé par [**SORTBY**]...[/**SORTBY**] pour trier la liste des fonctions.

Si on inclut encore des fichiers dans les fichiers inclus, on peut effectuer des tris sur des critères très sophistiqués.

[STRINDEX]...[/STRINDEX]

Recherche d'une sous-chaîne dans une chaîne

Syntaxe

```
[STRINDEX]pattern,n,string[/STRINDEX]
```

Description

On retourne la position d'une sous-chaîne `pattern` dans `string`, la recherche se fait à partir de la position `n` (0 étant la première position).

Si `n` est négatif, la recherche se fait à partir de la droite, -1 pour la première position à droite.

Si `pattern` contient des ",", il faut les précédés de "\".

`pattern` peut être une expression régulière.

Si non trouvé, on retourne -1

Exemple

[STRINDEX]cd,0,abcdefg[/STRINDEX] donne 2.

[STRINDEX]cd,1,abcdecdfg[/STRINDEX] donne 2.

La position du premier "cd" à partir de la position 1 est toujours 2.

[STRINDEX]cd,4,abcdecdfg[/STRINDEX] donne 5.

Le début de la recherche est 4, la position du premier "cd" à partir de la position 4 est 5.

[STRINDEX]cd,-1,abcdecdfg[/STRINDEX] donne 5.

La position du premier "cd" partant de la droite est 5.

[STRINDEX]cd,-5,abcdecdfg[/STRINDEX] donne 2.

La position du premier "cd" partant de la 5^{ème} position à partir de la droite est 2.

Expression régulière :

[STRINDEX]c.*0,abcd[/STRINDEX] donne 2. (c suivi de n'importe quel caractère).

Pattern avec ",":

[STRINDEX]C\,h,0,ABCDEc,hDFG[/STRINDEX] donne 5. On cherche la sous chaîne "c,h".

On peut combiner STRINDEX et SUBSTR pour extraire des sous-chaînes délimitées par des sous-chaînes:

Extraire la sous-chaîne avant "cd" dans "abcdefg":

```
[SV]string="abcdefg"[/SV]
```

```
[SV]i=[STRINDEX]cd,0,[GV]string[/GV][STRINDEX]+1[/SV]
```

```
[SUBSTR]0,[GV]i[/GV],[GV]string[/GV][SUBSTR]
```

cela donne "ab".

`[STRLEN]...[/STRLEN]`

Longueur de chaîne des caractères

Syntaxe

`[STRLEN]...[/STRLEN]` retourne la longueur de la chaîne des caractères contenue.

Description

`[STRLEN]abc[/STRLEN]` retourne 3.

`[STRLEN][AV]cn[/AV][[/STRLEN]` retourne la longueur de la valeur de cn.

[SUBSTR]n,m,string[/SUBSTR]

Description

Extraction d'une sous-chaîne

On extrait une sous chaîne dans string, à partir de la position n (0 étant la première position), une chaîne de longueur m.

Exemples

```
[SUBSTR]0,3,abcdefg[/SUBSTR] donne abc  
[SUBSTR]1,2,abcdefg[/SUBSTR] donne bc
```

Si les valeurs n et m ne sont pas corrects, le tag retourne une chaîne vide.

Bien sûr, [SUBSTR]...[/SUBSTR] peut contenir d'autres tags comme [AV], [GV], etc.

[TIMELIMIT] n[/TIMELIMIT]

Syntaxe

[TIMELIMIT]n[/TIMELIMIT]

Description

Paramètre de requête

Cette balise spécifie la durée maximum en seconde d'une recherche [LIST] ou [SEARCH]. Si à l'expiration de cette intervalle, la recherche ne s'est pas terminée, on reçoit une erreur TIMEOUT.

Cette erreur peut être capturée par :

[ON_ERROR LDAP_TIMEOUT]...[/ON_ERROR].

Comme pour les autres paramètres, si le modèle reçoit en argument TIMELIMIT=m, par exemple :

model.html?TIMELIMIT=10

La valeur de l'argument (10) emporte sur la valeur (n) spécifiée dans la balise [TIMELIMIT]n[/TIMELIMIT] à l'intérieur du modèle, si elle est présente.

En absence à la fois de l'argument et de la balise, c'est la variable SEARCH_TIMEOUT dans le fichier de configuration qui détermine la limite. Cette dernière est globale à tous les modèles.

Si null part la durée n'est spécifié, aucune limitation n'est imposée par WXD.

Dans tous les cas, le serveur LDAP peut imposer de son côté une limite.

Voir aussi

[SIZELIMIT]n[/SIZELIMIT]

[WHILE]condition[DO]list[/WHILE]]

Syntaxe

[WHILE]condition[DO]list[/WHILE]

Description

« condition » est la même que pour [IF]condition[THEN]...[/IF]

Tant que la condition est valide, on interprète list.

Il faut faire attention de ne pas boucler à l'infini.

4 Index

[TIMELIMIT] n/[TIMELIMIT]	109	[INLINE]	70
[ADD]	14	[INLINE][PARAM]params[/PARAM]contenu[/	
[ALI]...[/ALI]	17	INLINE]	70
[AN]	19	[INPUT_ATTR]...[/INPUT_ATTR]	73
[ARG]...[/ARG]	20	[LDAP_HOST]...[/LDAP_HOST]	74
[ATTACH]	94	[LDAP_PORT]...[/LDAP_PORT]	75
[ATTACH]...[/ATTACH]		[LIST]	76
Voir [SENDMAIL]	94	[MODIFY]	77
[ATTACHFILE]	94	[MODIFYRDN]	83
[ATTACHFILE]...[/ATTACHFILE]		[MSG]	94
Voir [SENDMAIL]	94	[MSG]	
[AUTH]	24	Voir [SENDMAIL]	94
[AV]...[/AV]	25	[MSGFILE]	94
[AVLI]...[/AVLI]	29	[MSGFILE]	
[BASE]...[/BASE]	31	Voir [SENDMAIL]	94
[CGL_ENV]...[/CGL_ENV]	32	[NB_CHILDREN]	85
[COMPARE]	33	[ON_ERROR]..[/ON_ERROR]	86
[COND]...[/COND]	34	[PARAM]params[/PARAM]	70
[COPY]	36	[PATH]	92
[CR]	37	[READ]	88
[CUR_DIR]	38	[REM]...[/REM]	89
[DAV]...[/DAV]	39	[REMOVE]...[/REMOVE]	90
[DAVLI]...[/DAVLI]	41	[REMOVE]...[/REMOVE]	92
[DECODE]...[/DECODE]	42	[RETURN_ATTRS]...[/RETURN_ATTRS]	91
[DELETE]	43	[RETURN_ATTRS]..[/RETURN_ATTRS]	91
[DEREF]x[/DEREF]	46	[SAVE][PATH]path[/PATH]contenu[/SAVE]	92
[DN]	44	[SAVE]...[/SAVE]	92
[ELI]...[/ELI]	49	[SEARCH]	93
[EMDL]...[/EMDL]	48	[SENDMAIL]	94
[ENC]...[/ENC]	50	[SENDMAIL][TO]...[FROM]...[SUBJECT]...[
[ENV]...[/ENV]	52	MSG]...[MSGFILE]...[ATTACH]...[/ATTA	
[FOOTER]...[/FOOTER]	53	CH][ATTACHFILE]...[/ATTACHFILE][[/SE	
[FROM]	94	NDMAIL]	94
[GET_MVAR <i>paramètre</i>]...[/GET_MVAR]	57	[SERVER_TIME]	96
[GETAFTER]...[/GETAFTER]	56	[SET_MVAR]...[/SET_MVAR]	97
[GETBEFORE]...[/GETBEFORE]	55	[SIZE_LIMITE_EXCEEDED]	101
[GV]...[/GV]	57	[SIZELIMIT] n/[SIZELIMIT]	100
[HEADER]...[/HEADER]	60	[SORTBY]...[/SORTBY]	102
[IF <i>paramètre</i>][THEN][ELSE] [/IF]	62	[STRINDEX]...[/STRINDEX]	105
[IF]...[ELSE]...[/IF]	62	[STRLEN]...[/STRLEN]	107
[IF][COND]...[/COND]...[ELSE]...[/IF]	34	[SUBJECT]	94
[INC <i>paramètre</i>]...[/INC]	66	[SUBSTR]n,m,string[/SUBSTR]	108
[INC]document[/INC]	66	[SV]...[/SV]	97

[TO]	94	liste	49
[V]	29	NAME	29
[WHILE]condition[DO]list[/WHILE]	110	NB_CHLD	62
AFLT	17	NB_RDN	27, 44
AFTER	21, 26, 44	NB_VAL	27
ALL	50	NEWDN	36
ALL_ARGS	67, 70	NEWRDN	83
ASCII	21, 27, 45	null	62
ATTRNAME	14, 77	PAT	55, 56
ATTRVAL	14, 77	pied de page	53
BEFORE	21, 26, 44	POST	21
BIN	25	PVAR	57
commentaire	89, 90	RDN	20, 25, 44, 83
commentaires	89, 90	RE_REPLACE	20, 25, 44
conditions	35	recherche	93
DEC	57	REPLACE	20, 25, 44
DECODE_URL	67, 70	RES	66
DESC	48	REVERSE	102
DN	14, 36, 77	SEP	29
DUMP	27	ST	25
EMPTY	67, 70	<i>submit</i>	14, 77
entête	60	supprimer	43
ENTRY_ATTR_MODEL_FILE	48	T61	27, 45
erreur	86	TEXT	67
FLT	49, 62	TEXT	67
FNAME	21	tri 102	
FORMAT	21, 26	UF	20, 25, 44
GET	21	UFN	20, 25
HIDDEN	102	UP	20, 25, 44
INDEX	26	Upload fichier	22
ISO8859	27, 45	URL	
LDAP_HOST	74	Modificateur	21, 26
LDAP_PORT	75	URL_DESC	21, 26
LE	27	UTCTIME	20, 26
LEAF	62	variable	57
LEN	44	<i>variable d'environnement</i>	42, 52
LG	25		