Le langage O_2shell

aide

help [*motclefO2*]

gestion de schémas et de bases

-create/delete schema/base *nomdelachose* /*La création permet de se positionner automatiquement dans le schéma ou la base créé.*/

-set schema/base *nomdelachose* /*positionnement dans un schéma ou une base*/

-create class *nomdelaclasse*
[inherit *nomdeclassemere*]
[public] type tuple (*attribut1* : *type1*,
attribut2 : *type2*,
...
attributn : *typen*)
[method public/private *nommethode*[(*paramètres*)] :*typeobtenu*,
nommethode2 ...,
...
end] /*la création des méthodes peut aussi se faire à part*/

-public/private method *nommethode*[(*paramètres*)] : *typeobtenu*,
in class *nomdeclasse*; /*La création de la méthode peut aussi se faire dans la classe*/

-method body *nommethode*[(*paramètres*)] : *typeobtenu*
in class *nomdeclasse*
{ *code*O_2C
return(*valeuraretourner*);
} /*Dans le corps, on peut faire référence à l'objet sur lequel est appelé la méthode grâce au mot clef **self**.*/

creation d'objets nommés

-name *nomdelobjet*

affichage

-display schemas/bases/classes/names/applications

-display schema/base/class/application *nomdelachose*

lancement d'OQL

-query

-quit

transactions

-abort /*annule une transaction*/

-commit /*confirme une transaction et libère la mémoire*/

-end transaction /*valide la transaction et passe en read-only*/

-quit /*libère la mémoire et termine la session*/

-transaction /* repasse en mode transaction depuis le mode read-only*/

-validate /*valide la transaction mais ne libère pas la mémoire et reste en mode transactionnel*/

confirmation des classes

-confirm classes

modification du statut de la base

-modify status TEST in base *nomdelabase*

nettoyage d'une base

`-garbage base` *nomdelabase*

Cette instruction permet d'éliminer les objets hors d'atteinte.

programmation

`-run body` {*code*O_2C }

`-run program` *nomduprog*[(*paramètres*)] `in application` *nomappli*

`-application` *nomappli*
`[variable` *nomvarglobale1* : *typevarglobale1* ;
nomvarglobale2 :*typevarglobale2* ;
...
`]`
`[program public/private` *nomprogramme1*,
`public/private` *nomprogramme2*,
...
`]`
`end`
/*Comme pour les méthodes on peut déclarer les programmes et les variables globales soit pendant la déclaration de l'application soit à part.*/

`-run application` *nomdelappli*

`-program body` *nomduprog*
`in application` *nomappli*
{ *code*O_2C } /*On peut utiliser des noms réservés pour nommer certains programmes : **init** (programme qui sera exécuté au lancement de l'application, on s'en sert généralement pour initialiser les variables globales), **exit** (sera exécuté à la sortie de l'application), **restart** (exécuté en cas d'arrêt du programme par la commande commit ou abort, ou en cas d'erreur ou d'interblocage).*/

`-display programs in application` *nomappli*

`-display program` *nomduprog* `in application` *nomappli*

`-display program doc` *nomduprog* `in application` *nomappli*

index

`-index` *nomdecollection* `on` *chemin*

`-display index [`*nomdecollection*`]` Affiche tous les index.

`-delete index` *nomdecollection* `on` *chemin*

cluster

`-cluster` *nomdeclasseoucollection* `[on` *chemin*`] [at` *priorité*`]`

`-delete cluster` *nomdelachose*

`-display clusters`