HDB-Queries

Objekt

@Klasse(Bedingung)

Bsp.

Gibt alle Objekte der angegebenen Klasse zurück, die in dem angegebenen Attribut den angegebenen Wert haben:

@Klasse(Attribut=Wert) -> @Account(Firstname=Hans)

Gibt alle Objekte der angegebenen Klasse zurück, die in den angegebenen Attributen die angegebenen Werte haben:

@Klasse((Attribut1=Wert) and (Attribut2=Wert)) -> @Account((Firstname=Hans) and (Lastname=Dampf))

Gibt alle Objekte der angegebenen Klasse zurück, die in den angegebenen Attributen den einen **oder** den anderen Wert haben:

@Klasse((Attribut1=Wert) or (Attribut2=Wert)) -> @Account((Firstname=Hans) or (Firstname=Tom))

Gibt alle Objekte der angegebenen Klasse zurück, die in dem angegebenen Attribut nicht leer sind / **irgendeinen** Wert haben:

@Klasse(Attribut=[something]) -> @Account(Firstname=[something])

Gibt alle Objekte der angegebenen Klasse zurück, die in dem angegebenen Attribut leer sind / **keinen** Wert haben:

@Klasse(Attribut=[nothing]) -> @Account(Firstname=[nothing])

Klassen unter einer Klasse

@Klasse(Bedingung)|Klasse(Bedingung)|Klasse(Bedingung)

Bsp.

Gibt alle Objekte der angegebenen Klasse2 an, die unter der angegebenen Klasse1 liegen und die in dem angegebenen Attribut den angegebenen Wert haben:

@Klasse1(Attribut=Wert)|Klasse2(*) ->@Mandator(cname=Q009)|MandatorAccountType(*)

Klasse nach ObjectId durchsuchen

Keine Verwendung der Contains Funktion:

Host/Partition/Folder/Share -> Man kann immer nur das Objekt welches direkt über der angegebenen ObjectId liegt herausbekommen

Verwendung der Contains Funktion:

Host/Partition/Folder/Share -> ObjectId des Shareobjekt in der Conatins Funktion verwenden -> gibt das Hostobjekt zurück

Bsp.

Gibt die angegebene darüber liegende Klasse zurück (ermöglicht es dazwischen liegende Klassen zu umgehen):

@Klasse(@Contains=ObjectId) -> @Host(@Contains=6073785)

HDB-Abfragen in VB ausführen

HDB-Abfragen in VB mit folgendem Befehl aufrufen:

Folgende Abfragen liefern eine String zurück:

oHDB.HDBQuery("HDB-Abfrage")

Attribut

@Klasse(Bedingung)|LDAP

Bsp.

Gibt das angegebene Attribut2, der angegebenen Klasse mit dem angegebenen Wert in dem angegebenen Attribut1 zurück:

Klasse(Attribut1=Wert)|Attribut2 -> Account(Firstname=Hans)|CName

Klassenhierarchie durchsuchen

Bsp.

Durchsucht die angegebene Klasse und die Klassen darüber nach dem angegebenen Attribut (Hört auf wenn ein Wert gefunden wurde. Existiert das Attribut in verschiedenen Objekten werden alle Werte von ~ getrennt ausgegeben):

Klasse(CName)|^Attribut -> MandatorAccountType(Admin)|^Account_Create_OULDAP

Klasse(CName)|Klasse(CName)|^Attribut -> Mandator(Q009)|MandatorAccountType(Admin)|^Account_Create_OULDAP

Durchsucht die angegebene Klasse und die Klassen darüer nach dem angegebenen Attribut (Listet alle Werte eines Objekts für das angegebene Attribut mit | getrennt auf. Existiert das Attribut in verschiedenen Objekten werden alle Werte von ~ getrennt ausgegeben):

Klasse(CName)|+Attribut ->MandatorAccountType(Admin)|^Account_Create_OULDAP

Klasse(CName)|Klasse(CName)|+Attribut -> Mandator(Q009)|MandatorAccountType(Admin)|+Account_Create_OULDAP

Folgende Abfragen liefern eine Objektliste zurück

Bsp.

Gibt eine Liste mit Objekten der angegebenen Klasse zurück, die in dem angegebenen Atrribut den angegebenen Wert haben:

oHDB.GetObjectList("Klasse", "Attribut = Wert") -> oHDB.GetObjectList("Account", "Firstname = Hans")

Gibt eine Liste mit Objekten zurück, die das angegebene SQL-Statement zurück gibt:

oHDB.GetObjectListSQL("SQL-Statement") -> oHDB.GetObjectListSQL("Select [id] from tblObject where Firstname = 'Hans' ")

Gibt eine Liste mit Objekten der angegebenen Klasse zurück. Als Bedingung kann man CNames, Objektpaths oder * verwenden.

oHDB.GetObjectListByPath("Klasse1(Bedingung)|Klasse2(Bedingung)") -> ohdb.getobjectlistbyPath("Mandator(Q009)|MandatorAccountType(*)")