Article
Eduard Lebedyuk · Sep 6, 2016 2m read

Some considerations when creating objects with >1 level of serial objects

Let's say we have two serial classes, one as a property of another:

```
Class test.Serial Extends %SerialObject
{
Property Serial2 As test.Serial2;
}

Class test.Serial2 Extends %SerialObject
{
Property Property As %String;
}
```

And a persistent class, that has a property of test.Serial type:

```
Class test.Persistent Extends %Persistent
{

Property Datatype As %String;

Property Serial As test.Serial;

}
```

So it's a serial, inside a serial, inside a persistent object.

When you have an object structure with more than one level of serial objects, it's important to remember, that ***if you modify a parent object via SQL, serial objects with serial objects would become initialized***.

Consider this method of test.Persistent class:

```
/// Do ##class(test.Persistent).Test()
ClassMethod Test()
{
    Do ..%KillExtent()
    Set Obj = ..%New()
    Set Obj.Datatype = 1
    Write $System.Status.GetErrorText(Obj.%Save())
    Kill

    Set Obj = ..%OpenId(1)
    Do Obj.Serial.%GetSwizzleObject(,.SerialBefore)
    Kill (SerialBefore)

    &sql(UPDATE test.Persistent SET Datatype = 2)
```

```
    Set Obj = ..%OpenId(1)
    Do Obj.Serial.%GetSwizzleObject(,.SerialAfter)

    Zw SerialBefore,SerialAfter
}
```

In this method we compare serialization of serial object with a serial object before and after the main persistent object gets updated via SQL. Here's the output:

```
>Do ##class(test.Persistent).Test()
SerialBefore=""
SerialAfter=$lb($lb($lb("")),"test.Serial")
```

Note, that while they are the same objects (all properties are empty), the serialization changes.

The code on GitHub.

I would like to thank Dmitry Zasypkin and Alexander Koblov for their help in figuring this out.

#Object Data Model #SQL #Caché