

Caché Objects QuickStart

Москва, 24 марта 2003

Copyright © InterSystems, 2003

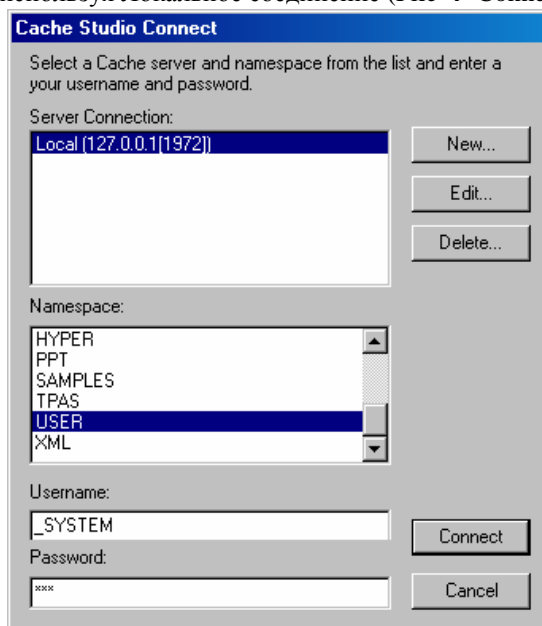
Оглавление

Создание класса	3
Работа с объектами в Caché Object Script	5
Создание и работа с запросами	6
Создание SQL View	7
Экспорт данных в XML-файл	7
Связанные классы	8
Наследование	11
Коллекции	13
Отношения	14
Сервер Caché Object для ActiveX	16
Создание проекта Visual Basic	16
Класс Factory	17
Соединение с сервером	17
Открытие существующего объекта Caché	18
Класс ResultSet	18
Мастер форм Caché (CachéForm Wizard)	18
Что дальше?	19
Приложение 1	19
Приложение 2 (ClassDefinition)	20
Приложение 3 (Работа с Delphi)	20

Создание класса

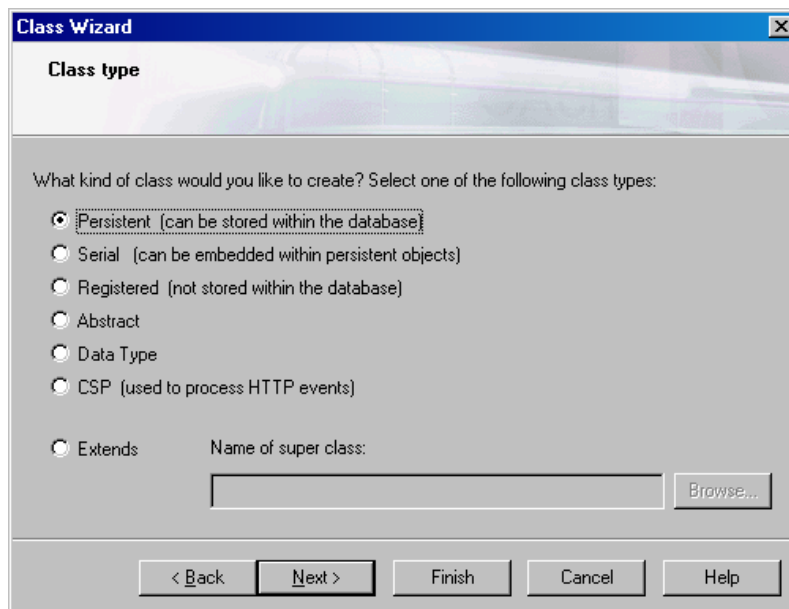
Для создания и редактирования классов в Cache предназначена утилита Cache Studio.

Подключитесь к Cache, используя Локальное соединение (File → Connect).



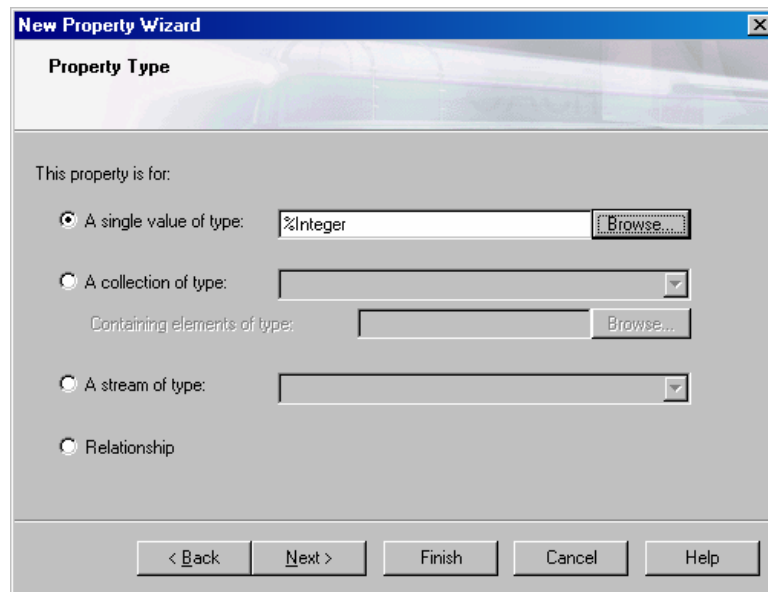
Создайте новый проект (File → New Project) и сохраните его под именем First (File → Save Project). Проект Cache Studio объединяет классы, программы на CacheObjectScript, Cache Basic и страницы Cache Server Pages.

Создайте новый класс Human (File → New → Cache Class Definition) типа Persistent.



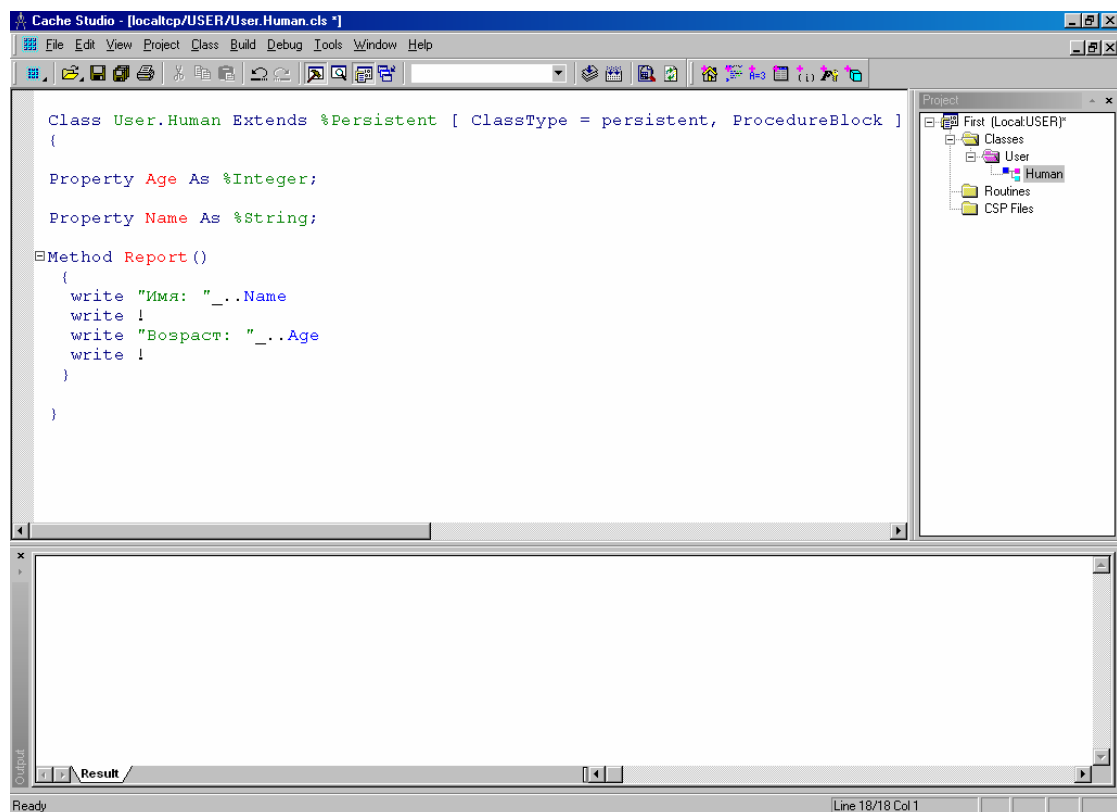
Добавьте к классу Human два свойства (Class → Add → New Property):

- Name с типом %String
- Age с типом Integer



Добавьте к классу Human метод Report, не имеющий аргументов и не возвращающий никакого значения.

Обратите внимание, что каждой строчке предшествует как минимум один пробел.



Добавьте индекс по свойству Name.

Скомпилируйте класс (Ctrl + F7).

Простейший класс Human с двумя свойствами и одним методом готов.

Работа с объектами в Caché Object Script

Войдите в Caché Terminal.

Создайте новый экземпляр объекта Human – объект h

```
USER>set h=##class(User.Human).%New()
```

Определите свойства объекта

```
USER>Set h.Name="Иван"
```

```
USER>Set h.Age=11
```

Выполните метод Report объекта h

```
USER>do h.Report()
```

```
Имя: Иван
```

```
Возраст: 11
```

Сохраните и закройте объект h

```
USER>do h.%Save()
```

```
USER>kill h
```

Таким же образом создайте и сохраните еще два объекта типа Human, например, Петр – 12 лет и Василий – 10 лет.

Откройте один из созданных объектов

```
USER>set h=##class(User.Human).%OpenId(2)
```

Проверьте значения его свойств:

```
USER>write h.Name
```

```
Петр
```

```
USER>write h.Age
```

```
10
```

Удалите из базы этот объект.

```
USER>do ##class(User.Human).%Delete(h.%Oid())
```

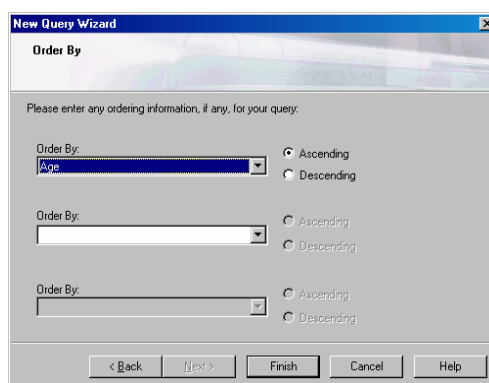
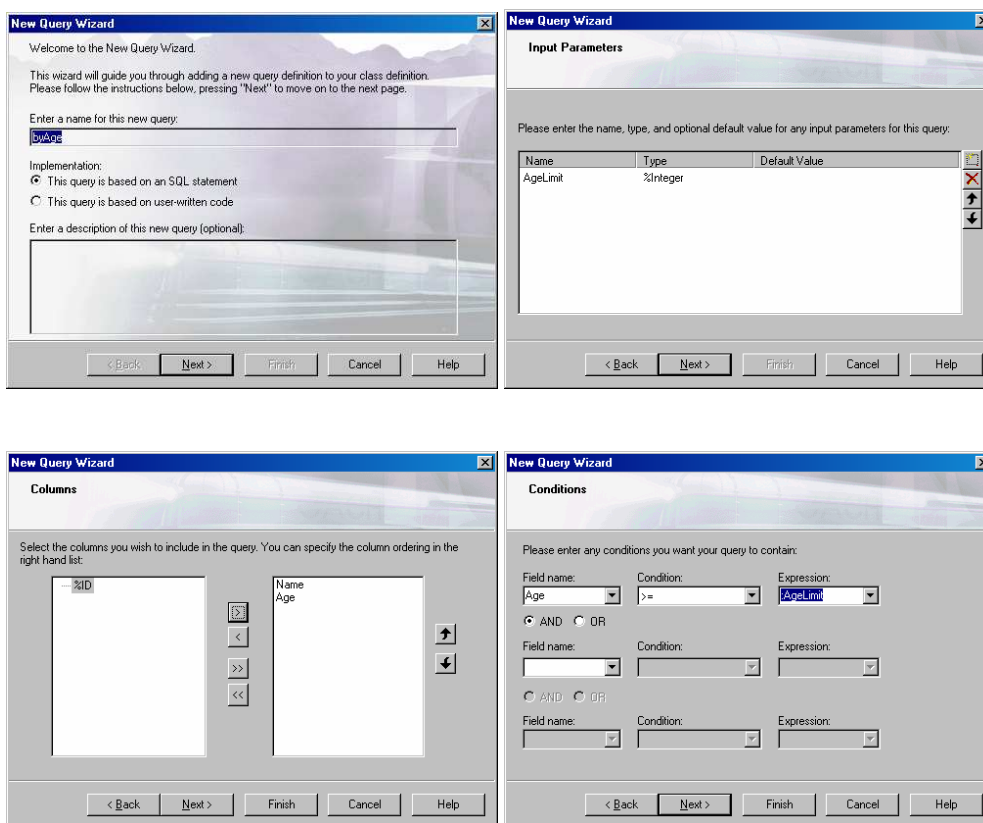
Данная операция удаляет объект из базы, но не из памяти. Поэтому необходимо закрыть экземпляр объекта в памяти, не сохраняя его на диске.

```
USER>kill h
```

Создание и работа с запросами

Создайте в классе Human при помощи мастера запросов запрос byAge. Запрос будет основан на SQL-утверждении (Implementation: This Query is based on SQL statement).

Запрос должен принимать один параметр AgeLimit, типа %Integer, и возвращать поля Name и Age, при условии, что поле Age объекта больше или равно значению параметра AgeLimit. Возвращаемые запросом объекты должны быть отсортированы по полю Age.



Скопируйте класс.

В Cache Script работа с запросами выглядит следующим образом:

Открытие запроса:

```
USER>set rs=##class(%ResultSet).%New("User.Human.byAge")
```

Инициализация запроса:

```
USER>do rs.Execute(10)
```

Переход на следующую строку результата:

```
USER>write rs.Next()
```

```
1
```

1 – данные присутствуют, 0 – данных больше нет.

Извлечение данных:

```
USER>write rs.Get("Name")
```

```
Петр
```

Закрытие запроса:

```
USER>kill rs
```

Caché Terminal позволяет выполнять запрос в режиме командной строки. Для создания программ создадим в нашем проекте простейшую программу (File → New → Cache ObjectScript Routine), которая выводит результаты запроса на экран:

```
QueryHuman(AgeLimit)
```

```
new rs
set rs=##class(%ResultSet).%New("User.Human.byAge")
do rs.Execute(AgeLimit)
while rs.Next() {
    write "Human",!
    write rs.Get("Name")_" " "_rs.Get("Age")
    write ! }
quit
```

Вы должны сохранить программу в проекте First под именем QueryHuman.MAC и откомпилировать её. Результат выполнения этой программы показан ниже:

```
USER>do ^QueryHuman(10)
```

```
Human
```

```
Петр 12
```

```
Иван 22
```

Создание SQL View

Создайте новый запрос класса без параметров. Объявите его SQLView. Обратитесь к этому виду через SQL Manager.

Экспорт данных в XML-файл

В Caché Studio в окне Inspector добавьте классу User.Human суперкласс %XML.Adaptor

Создайте метод класса для экспорта данных в XML-файл:

```

ClassMethod Export(file As %String) As %Status
{
  open file:"WNS"
  use file
  write "<!DOCTYPE Humans [",!
  do ##class(User.Human).XMLDTD()
  write "]>",!
  write "<Humans>",!
  set rs=##class(%ResultSet).%New("Human:Extent")
  set rc=rs.Execute()
  for {
    quit:'rs.Next()
    set id=rs.GetData(1)
    set human=##class(Human).%OpenId(id)
    do human.XMLExport()
  }
  write "</Humans>",!
  close file
  quit $$$OK
}

```

Запустите метод из Терминала

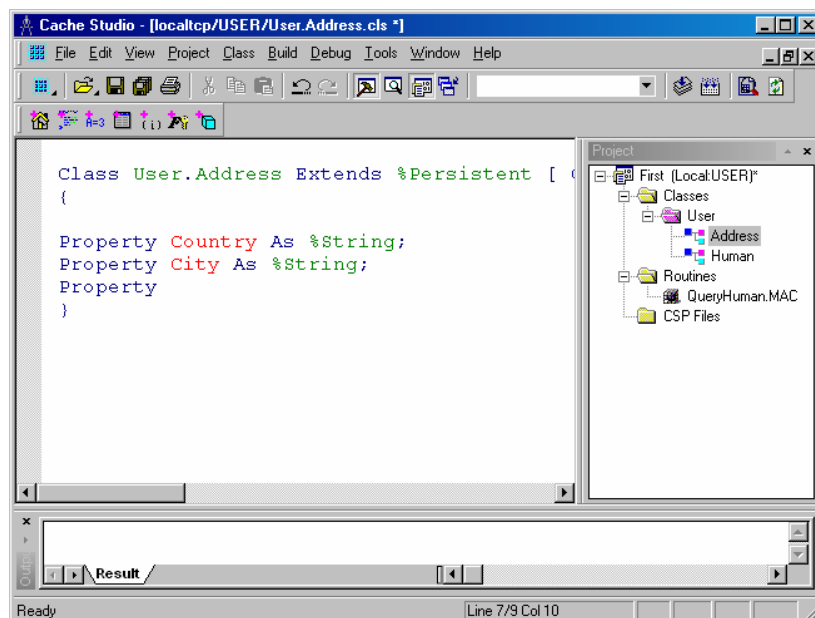
```
USER>Do ##class(User.Human).Export("c:/humans.xml")
```

Присвойте значение Attribute параметру XMLPROJECTION свойства Name.

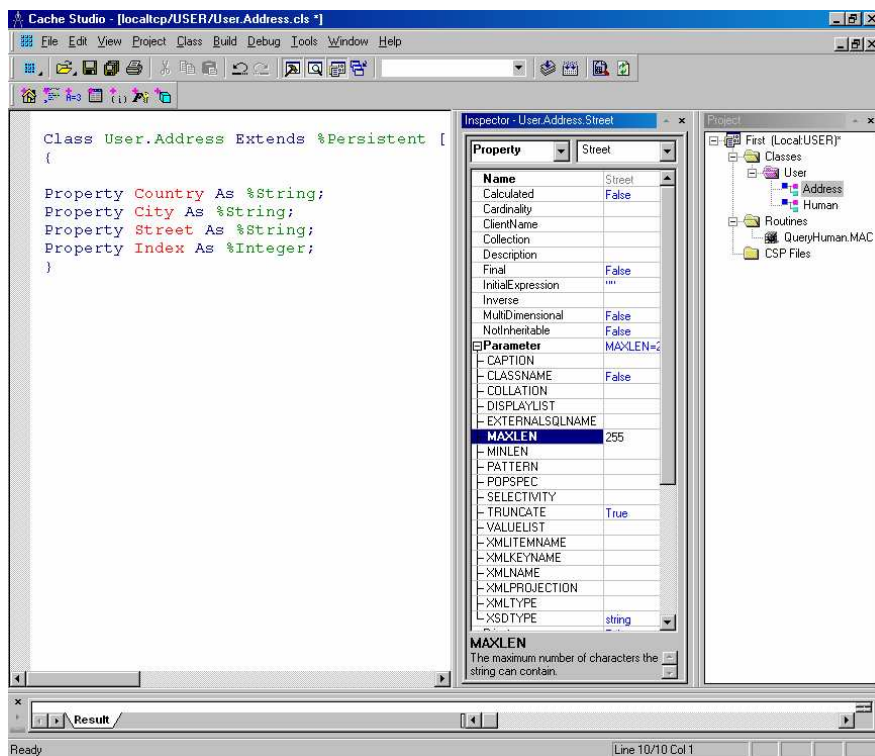
Запустите метод еще раз.

Связанные классы

Создайте Persistent класс Address с атрибутами Country, City, Street и Index, причем для атрибута Street укажите максимальную длину 255. Вы можете добавлять свойства с помощью соответствующего Мастера или вручную.

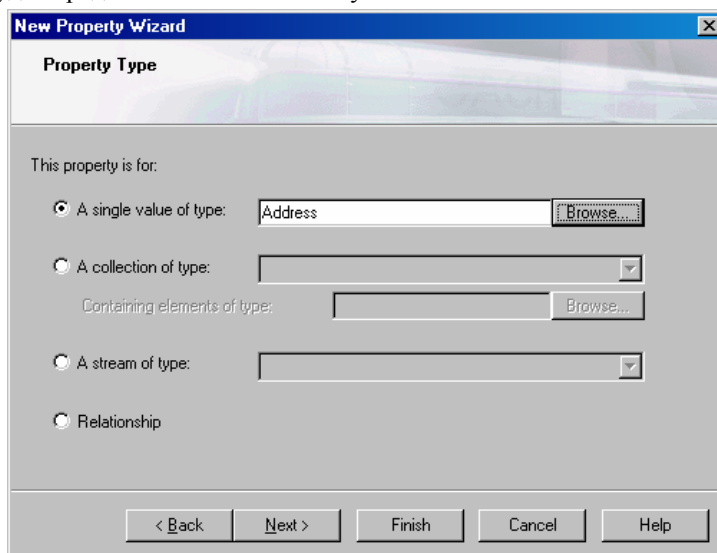


Для свойства Street в окне Inspector задайте значение параметра MAXLEN = 255:



Скомпилируйте класс.

Добавьте в класс Human свойство Home, типом которого будет являться Address, а поскольку Address сам является объектным классом (а не литеральным, как, например, %Integer), то атрибут Home будет представлять собой ссылку на объект типа Address.



Скомпилируйте класс Human.

В Caché Terminal создайте объекты *h* и *a*, классов *Human* и *Address* соответственно и заполните их свойства.

```
USER>set h=##class(User.Human).%New()  
USER>set h.Name="Евгений"  
USER>set h.Age=55  
  
USER>set a=##class(User.Address).%New()  
USER>set a.Country="Россия"  
USER>set a.City="Москва"  
USER>set a.Index=121019  
USER>set a.Street="Волжонка 6, офис 14"
```

Присвойте атрибуту `Home` объекта `h` значение – ссылку на объект `a`:

```
USER>set h.Home=a
```

Теперь все свойства объекта `a` становятся доступны через ссылку `h.Home`

```
USER>write h.Home.Street  
Волжонка 6, офис 14
```

Сохраните и закройте объект `a`.

```
USER>do a.%Save()  
USER>kill a
```

Несмотря на то, что объект `a` больше не находится в памяти, Вы по-прежнему можете обратиться к нему через ссылку `h.Home`:

```
USER>set h.Home.Country="Российская Федерация"  
  
USER>write h.Home.Country  
Российская Федерация
```

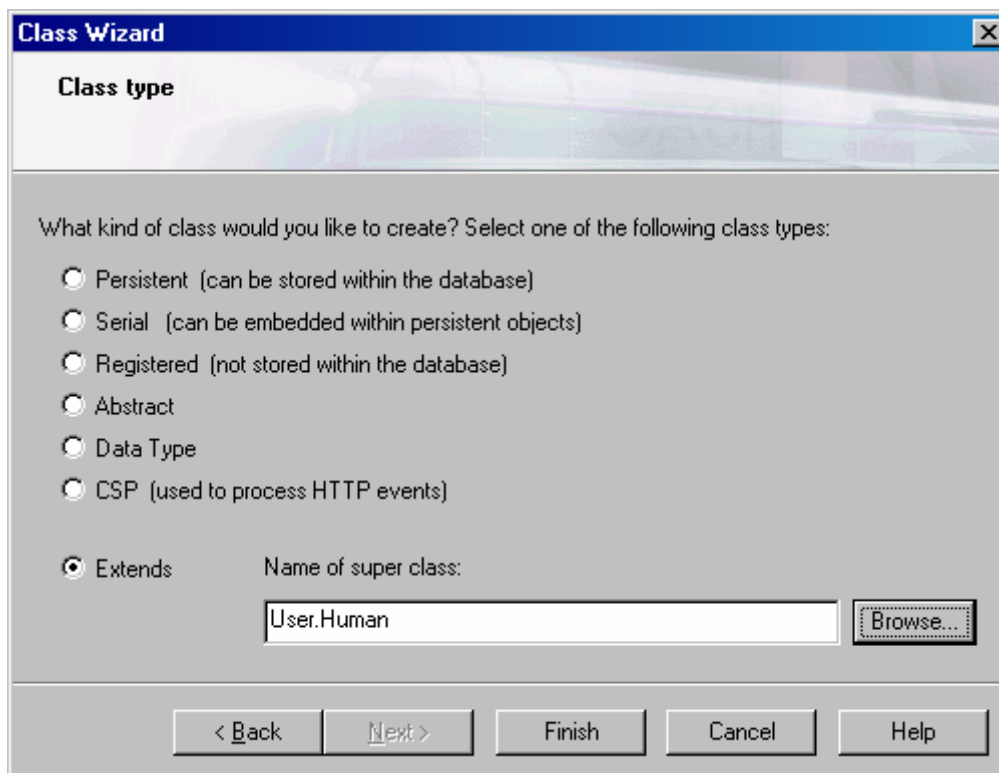
Технология, позволяющая по ссылке обращаться к свойствам и методам объекта, не выполняя дополнительных операций по его открытию и закрытию, называется Свиззлинг (Swizzling).

Сохраните и закройте объект `h`.

```
USER>do h.%Save()  
USER>kill h
```

Наследование

Создайте класс Patient, унаследованный от Human



Класс Patient унаследовал от Human все свои свойства и методы. Добавьте в класс Patient новое свойство Diagnosis типа %String и переопределите метод Report таким образом, чтобы он показывал и диагноз:

```
write "Имя: " ..Name
write !
write "Возраст: " ..Age
write !
write "Диагноз: " ..Diagnosis
quit
```

В Caché Terminal создайте экземпляр класса пациент и сохраните его.

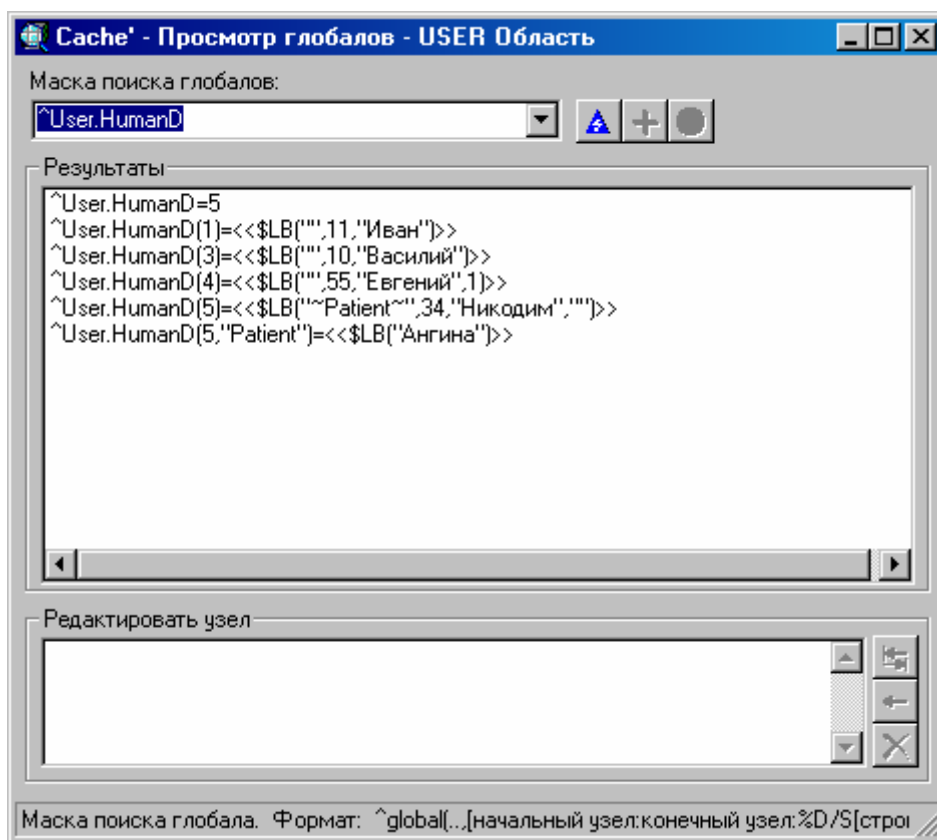
```
USER>set p=##class(User.Patient).%New()
USER>set p.Name="Никодим"
USER>set p.Age=34
USER>set p.Diagnosis="Ангина"

USER>do p.Report()
Имя: Никодим
Возраст: 34
Диагноз: Ангина
```

```
USER>do p.%Save()
```

```
USER>kill p
```

В Проводнике Cache посмотрите на глобаль ^User.HumanD области USER.



По структуре глобали видно, что в ней содержатся данные объектов класса Human и унаследованных от него классов. Также видно, что объекты класса и его подклассов имеют общую сквозную нумерацию, а последний сохраненный вами объект класса Patient имеет ID=5.

Поскольку класс Patient унаследован от класса Human, то любой объект класса Patient является также и объектом класса Human. Откройте объект с ID=5 как объект класса Human:

```
USER>set h=##class(User.Human).%OpenId(5)
```

```
USER>do h.Report()
```

Имя: Никодим

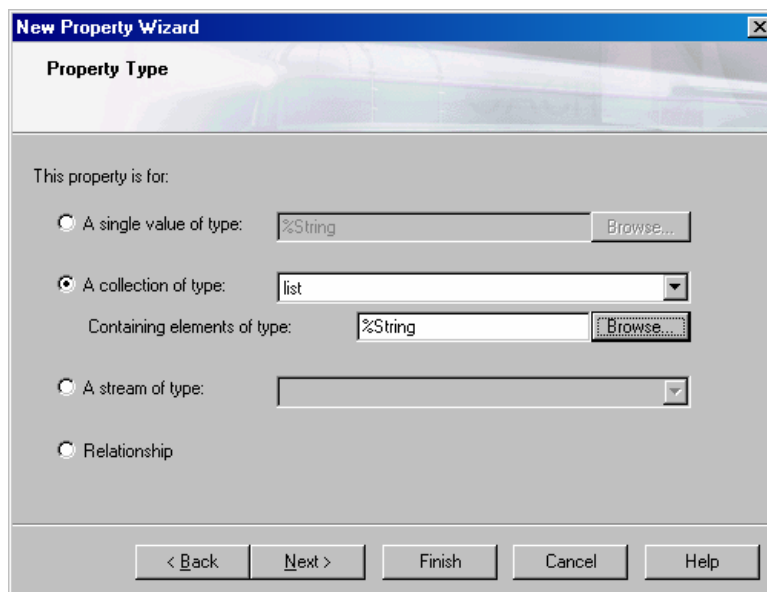
Возраст: 34

Диагноз: Ангина

Как видно, объект класса Patient может быть адресован как объект своего родительского класса, но при этом продолжает себя вести как объект класса, в качестве экземпляра которого он был создан.

Коллекции

Добавьте к классу Patient свойство Symptoms типа %String и являющееся коллекцией типа список (List).



В Caché Terminal откройте существующий объект класса Patient.

```
USER>set p=##class(User.Patient).%OpenId(5)
```

Вставка элементов в конец списка:

```
USER>do p.Symptoms.Insert("Кашель")
USER>do p.Symptoms.Insert("Насморк")
```

Вставка элемента в конкретное место списка:

```
USER>do p.Symptoms.InsertAt("Жар",2)
```

Количество элементов в списке:

```
USER>write p.Symptoms.Count()
3
```

Получение значения конкретного элемента:

```
USER>write p.Symptoms.GetAt(2)
Жар
```

Изменение значения элемента списка:

```
USER>do p.Symptoms.SetAt("Температура",2)
```

Удаление элемента из списка:

```
USER>do p.Symptoms.RemoveAt(2)
```

Сохраните и закройте объект.

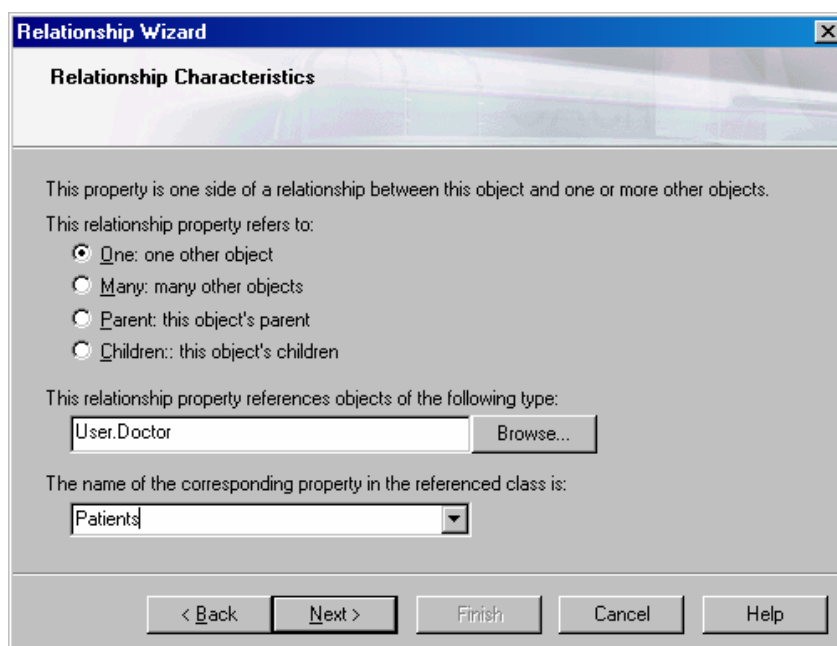
Отношения

Создайте Persistent-класс Doctor со свойством Name.

Предположим, доктор может одновременно лечить нескольких пациентов, в то время как один пациент может наблюдаться только у одного доктора. В этом случае наши классы объединены отношением типа Один-ко-Многим.

У пациента создайте поле Physician, которое будет ссылкой на объект класса Doctor и объявите ее отношением типа один ко многим с мощностью связи поля Physician – One.

Укажите имя связанного свойства со стороны класса Doctor – Patients. Это свойство будет определять отношение в классе Doctor.



Скомпилируйте оба класса.

Создайте в памяти двух пациентов и доктора:

```
USER>set p1=##class(Patient).%New()
```

```
USER>set p1.Name="Иван"
```

```

USER>set p2=##class(Patient).%New()
USER>set p2.Name="Петр"
USER>set d=##class(User.Doctor).%New()
USER>set d.Name="Профессор Иванов"

```

Свяжите пациентов и доктора двумя возможными способами:

```

USER>set p1.Physician=d
USER>do d.Patients.Insert(p2)

```

Теперь Вы можете получать информацию о связанных объектах с обеих сторон отношения:

```

USER>w d.Patients.GetAt(1).Name
Иван
USER>w d.Patients.GetAt(2).Name
Петр
USER>w p1.Physician.Name
Профессор Иванов

```

Как видите, со стороны Many (Doctor) работа с отношением аналогична работе с коллекцией, а со стороны One с обычной ссылкой.

Сохраните доктора и посмотрите на глобалы, соответствующие обоим классам.

```

USER>do d.%Save()

```

Как видите, сохранение доктора привело и к сохранению пациентов.

Теперь закройте все объекты и попытайтесь удалить объект Doctor с ID=1.

```

USER>w ##class(User.Doctor).%DeleteId(1)
0  0;0User.Patient.Physician

USER>do $system.OBJ.DisplayError()
ERROR #5823: Cannot delete object, referenced by
'User.Patient.Physician'

```

Как видите, операция не удалась. В отношении типа Один-ко-Многим автоматически поддерживается ссылочная целостность. Пока Вы не удалите (переключите на другого доктора) всех пациентов конкретного доктора, Вы не сможете удалить доктора из базы.

Попробуйте реализовать отношение типа Parent-Child на примере классов Book и Chapter.

Обратите внимание, на то, что экземпляры Chapter хранятся в глобали класса Book, а удаление объекта Book приводит к удалению всех связанных с ним объектов класса Chapter.

Сервер Caché Object для ActiveX

Интерфейс Caché Objects с ActiveX позволяет организовать взаимодействие многочисленных клиентских приложений и инструментальных средств разработки (Visual Basic, Delphi, C++ Builder) с объектами Caché.

Caché Objects состоит из следующих компонентов ActiveX:

- *Сервер Caché Object для ActiveX*. Сервер автоматизации (automation server), представляющий объекты Caché в виде объектов ActiveX.
- *Caché Object Form Wizard* (мастер форм Caché). Дополнительно подключаемый модуль (Add-In) для Visual Basic, позволяющий автоматически создавать формы для быстрого доступа к объектам Caché.

Сервер Caché Object для ActiveX обеспечивает удобный доступ к объектам Caché любым приложениям, поддерживающим объекты ActiveX.

Сервер Caché Object для ActiveX – это библиотека CacheObject.dll, которая содержит шесть типов ActiveX объектов:

1. *Caché Factory* – фабрика объектов Caché.
2. *Caché ObjInstance* – класс клиентских объектов, соответствующих объектам сервера Caché
3. *Caché ResultSet* – интерфейс для выполнения запросов.
4. *Caché SysList* – класс, предназначенный для обработки данных, имеющих формат списка Caché (\$LB).
5. *Caché BinaryStream* – интерфейс для использования потоков данных Cache.
6. *Caché CharStream* - интерфейс для использования потоков данных Cache.

В данном документе рассматривается использование первых трех типов ActiveX объектов.

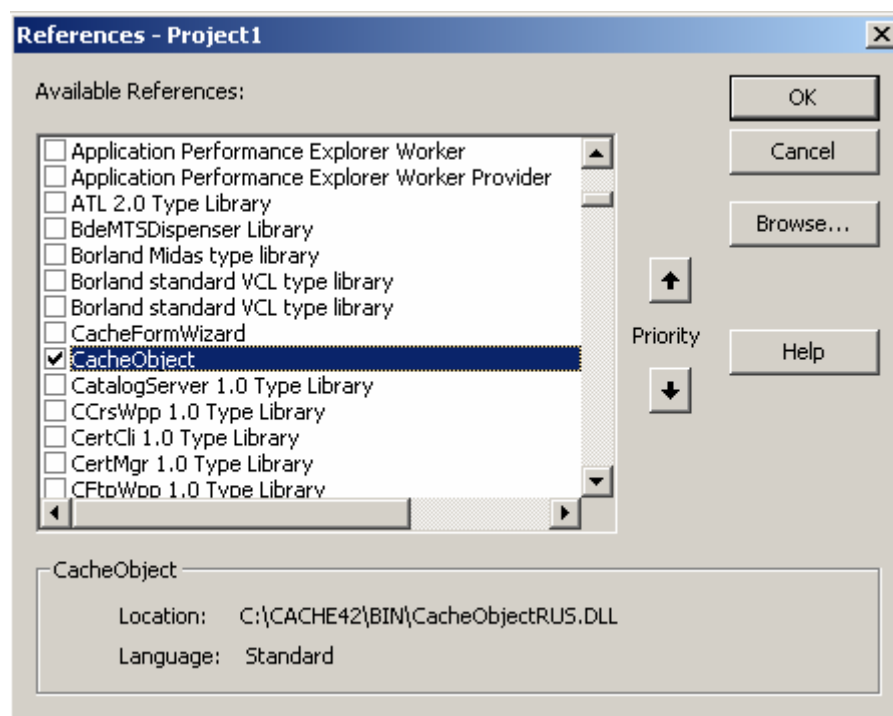
В качестве примера для изучения ActiveX интерфейса Caché рассмотрим работу с Visual Basic.

Создание проекта Visual Basic

Создайте новый стандартный проект Visual Basic.

Вы должны добавить в проект ссылку на CacheObject:

1. Выберите библиотеку CacheObject (через опцию меню **Project | References**)



Если Вы не нашли библиотеку CacheObject (как правило она устанавливается автоматически), Вы должны её добавить.

2. Щелкните по кнопке Browse...
3. Выберите файл CacheObject.dll, отыскав его по вложенной папке Вашей инсталляции Caché (обычно \CacheSys\bin) и открыв щелчком по кнопке Open.

Далее в ходе создания учебного проекта будет рассмотрено использование классов Factory и ResultSet. Кроме этого, будет описана работа с Мастером форм Caché (CacheForm Wizard).

Класс Factory

Класс Factory предназначен для установления и управления соединением с сервером Cache. Кроме этого, этот класс служит «фабрикой объектов» для создания экземпляров новых объектов ActiveX типа ObjInstance: либо путем открытия существующих, либо путем создания новых объектов Caché.

Класс Factory располагает следующими основными методами:

- метод Connect() устанавливает соединение с сервером Cache;
- метод ConnectDlg() открывает окно диалога для задания параметров соединения;
- метод Disconnect() разрывает соединение с сервером Cache;
- метод IsConnected() проверяет, существует ли соединение с сервером Cache;
- метод New() создает новый объект ActiveX;
- метод OpenId() открывает существующий объект, используя имя класса и ID;

Соединение с сервером

Для того чтобы установить соединение с сервером Caché наберем следующий код:

Option Explicit

Dim Factory As CacheObject.Factory

Dim connectstring As String

Dim success As Boolean

Private Sub Form_Load()

`Создаем экземпляр класса Factory

`Для присваивания объекту значения используется Set

Set Factory = CreateObject("CacheObject.Factory")

`Если соединение с сервером отсутствует, создаем его

If Not Factory.IsConnected() Then

`Строка параметров соединения может быть задана явно:

connectstring = "cn_iptcp:127.0.0.1[1972]:USER"

`В качестве альтернативы можно использовать диалоговое окно:

`connectstring = Factory.ConnectDlg()

success = Factory.Connect(connectstring)

End If

End Sub

Открытие существующего объекта Cache

После соединения с сервером Cache откроем объект класса User.Human с Id=1 и выведем на экран значения свойств Name и Age этого объекта.

Для этого воспользуемся методом Open объекта Cache Factory:

```
Dim human As CacheObject.objinstance
Set human = factory.OpenId("User.Human",1)
```

Далее Вы получаете доступ к свойствам и методам этого объекта.

```
MsgBox (human.Name + " " + human.Age)
```

Дополнительную информацию об интерфейсе Cache Object для ActiveX Вы можете получить в документации Cache в разделах Cache ActiveX Class Reference и Developing with Cache Objects (глава 11 «The ActiveX Binding»).

Класс ResultSet

Класс ResultSet представляет собой интерфейс для выполнения запросов. Каждый объект ResultSet связывается с конкретным запросом из определения класса Cache. Объект ResultSet позволяет не только выполнить запрос, но и проанализировать результат. Объект ResultSet располагает следующими основными методами:

- метод Execute() выполняет запрос, с которым связан объект ResultSet;
- метод Next() осуществляет переход на следующую строку выборки;
- метод Get() возвращает значение из заданного поля (столбца);
- метод Close() закрывает ResultSet после завершения его использования.

Кроме этого Вы можете получить информацию о запросе с помощью методов GetParamCount(), GetParamName(), GetColumnCount() и GetColumnName().

Усовершенствуйте Ваш проект. Добавьте компонент ListBox и по какому-либо событию вызовете следующий код:

```
Dim rs As CacheObject.ResultSet
Set rs = Factory.ResultSet("User.Human", "byAge")
rs.Execute (10)
While rs.Next()
    List1.AddItem rs.Get("Name") + " " + rs.Get("Age")
Wend
rs.Close
```

В ListBox будет добавлена информация об объектах класса User.Person.

Мастер форм Cache (CacheForm Wizard)

Мастер форм Cache представляет собой простой в применении дополнительно подключаемый модуль Visual Basic (Add-In). С его помощью можно автоматически генерировать формы Visual Basic, позволяющие просматривать, изменять или добавлять в БД экземпляры класса. При этом имеется возможность выбрать содержащиеся в форме свойства классов объектов и связанные классы, а также выполнить поиск с использованием запросов, определенных для класса.

Для того чтобы воспользоваться Мастером форм Cache:

1. Создайте новый проект Visual Basic.

2. Выберите CacheForm Wizard (через опцию меню Add-Ins | CacheForm Wizard).
3. Выберите Namespace «USER».
4. Выберите класс «User.Human».
5. Выберите свойства класса «User.Human» и связанных с ним классов, которые Вы хотите поместить на форму.
6. Установите заголовок на форме для каждого свойства класса (по умолчанию он совпадает с названием соответствующего свойства).
7. Нажмите кнопку «Закончить»
8. Запустите проект

С помощью Мастера форм Cache Вы создали простейшее приложение на Visual Basic, взаимодействующее с сервером Cache с помощью интерфейса Cache Object для ActiveX.

В коде форм, созданных Мастером форм Cache, используется Cache Factory для соединения с сервером и получения доступа к объектам Cache. Сгенерированный код вполне может служить образцом для самостоятельного программирования взаимодействия GUI-приложений с сервером Cache Object для ActiveX.

Что дальше?

Если Вы успешно прошли все этапы этого курса, Вы имеете представление об объектах Cache и о работе с ними через ActiveX интерфейс. При помощи документации Cache и дополнительных источников информации, указанных в Приложении, рекомендуем Вам самостоятельно изучить следующие темы:

- Классы BinaryStream и CharStream
- Класс Cache SysList
- Использование объектов Cache при разработке приложений на Visual Basic
- Использование объектов Cache при разработке приложений на Delphi (см. [Приложение 3](#))
- Работа с потоками из Cache ObjectScript

Приложение 1

Для изучения Cache Вам будут полезны следующие разделы документации:

- Using Cache Studio
- Cache Objects Programmer's Guide

Кроме документации для изучения Cache можно порекомендовать следующие материалы и книги:

1. CSP QuickStart. QuickStart поможет изучить Cache Server Pages – технологию создания Web-приложений.
2. Материалы учебного курса Pattaya. Учебный курс проводится консультантами InterSystems в Таиланде. Материалы курса содержат примеры приложений на Visual Basic, Delphi, Java, C++, Cache Server Pages. (<ftp://ftp.intersys.com/pub/Pattaya/Pattaya50.zip>)
3. Кирстен В., Иренгер И., Рёриг Б., Шульте П. СУБД Cache: объектно-ориентированная разработка приложений. - СПб, «Питер», 2001.
4. Кречетов Н., Петухова Е., Скворцов В., Умников А., Шукин Б. Постреляционная технология Cache' для реализации объектных приложений. –М, МИФИ, 2001

Дополнительные материалы Вы можете найти в разделе Разработчику сайта Московского представительства InterSystems Corp. (www.intersystems.ru) и в файловых архивах конференции

CACHE_RU (http://groups.yahoo.com/group/cache_ru/files/). Дополнительную информацию Вы можете получить в российском представительстве InterSystems Corp.

Приложение 2 (ClassDefinition)

Часто возникает необходимость получить доступ к метаданным Cache. Для этого используются системные классы пакета %Dictionary, которые позволяют Вам просматривать информацию о классах Cache, их свойствах, методах, параметрах, запросах, создавать новые классы, свойства, методы и т.д.

Вы можете получить подробную информацию об этом пакете в разделе Class Definition Classes руководства Cache Objects Programmer's Guide.

Приложение 3 (Работа с Delphi)

Вы можете использовать сервер Cache Object для ActiveX при создании приложений на Delphi. Вы работаете с объектами Cache в Delphi точно так же, как в VB.

В представленной ниже программе на Delphi показана работа с классами Factory и ResultSet:

```
unit Example;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ComObj, StdCtrls;
type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    ListBox1: TListBox;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  Factory : Variant;
  Human : Variant;
implementation
procedure TForm1.FormCreate(Sender: TObject);
var
  ConnectStr : String;
begin
  Factory := CreateOleObject('CacheObject.Factory');
  ConnectStr := Factory.ConnectDlg;
```

```
    If Not Factory.Connect(ConnectStr)
    then MessageDlg('Could not connect to Caché',mtError,[mbok],0);
    Human := Factory.OpenId('User.Human',1);
    MessageDlg(Human.Name+' '+inttostr(Human.Age), mtInformation,[mbOk], 0);
end;
procedure TForm1.Button1Click(Sender: TObject);
var
    resultset : variant;
begin
    resultset := factory.resultset('User.Human','byAge');
    resultset.Execute(10);
    While resultset.Next Do
    begin
        ListBox1.Items.Add(resultset.get('Name')+' '+
inttostr(resultset.getdata('Age')));
    end;
    resultset.close;
end;
end.
```