13. ИНТЕРАКТИВНАЯ РАБОТА С ПОЛЬЗОВАТЕЛЕМ

Рассмотрим некоторые вопросы интерактивной работы пользователя в условиях выключенной модальности (в свойство конфигурации *Режим использования модальности* установлено значение *Не использовать*).

Все методы интерактивной работы пользователя функционируют только в клиентском контексте.

1. ПоказатьПредупреждение

- самая простая процедура интерактивной работы с пользователем, это аналог метода – Предупреждение (работает в режиме модальности) (см. рис. 13.1).

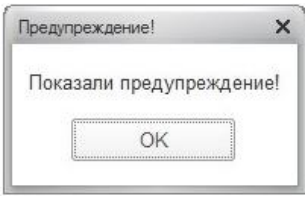


Рис. 13.1. Окно предупреждения

Синтаксис метода «ПоказатьПредупреждение»:

ПоказатьПредупреждение(<ОписаниеОповещенияОЗавершении>, <ТекстПредупреждения>, <Таймаут>, <Заголовок>)

Где:

ОписаниеОповещенияОЗавершение – параметр имеет тип «ОписаниеОповещения», в котором содержится название экспортной процедуры, выполняемой после нажатия кнопки «ОК» окна предупреждения, этот параметр не обязательный;

ТекстПредупреждения – текст самого предупреждения (тип *Строка* или *ФорматированнаяСтрока*);

Таймаут – интервала времени в секундах, в течение которого будет отображаться окно предупреждения. Если параметр не указан, то время не ограничено.

Заголовок – заголовок окна предупреждения.

Пример 1.

Вывести предупреждение пользователю.

1) Создать обработку, форму обработки, добавить команду формы *ПоказываемПредупреждение* и разместить ее на форме
2) В обработчике команды записать следующий код:

```
&НаКлиенте
Процедура ПоказываемПредупреждение(Команда)
    ПоказатьПредупреждение(,"Показали предупреждение!",,"Предупреждение!");
КонецПроцедуры
```

Листинг 13.1. Вывод простого предупреждения

3) Проверить работу команды в пользовательском режиме (должно выйти окно предупреждения, как на рисунке 13.1).

Рассмотрим пример работы метода **ПоказатьПредупреждения** с первым параметром – ОписаниеОповещенияОЗавершении, указывающем процедуру, которая сработает после нажатия кнопки «ОК» окна предупреждения (см. листинг 13.2).

4) Изменить программный код:

```
&НаКлиенте
Процедура ПоказываемПредупреждение(Команда)
      ОписаниеОповещения = Новый
      ОписаниеОповещения("ПослеЗакрытияПредупреждения",ЭтаФорма);
     ПоказатьПредупреждение(ОписаниеОповещения,"Показали предупреждение!",,"Предупреждение!");
КонецПроцедуры

&НаКлиенте
Процедура ПослеЗакрытияПредупреждения(Параметры) Экспорт
    Сообщить("Закрыли окно предупреждения");
КонецПроцедуры
```

Листинг 13.2. Использование описания оповещения

5) Проверить работу команды в пользовательском режиме (после нажатия кнопки ОК окна предупреждения должно выйти сообщение.

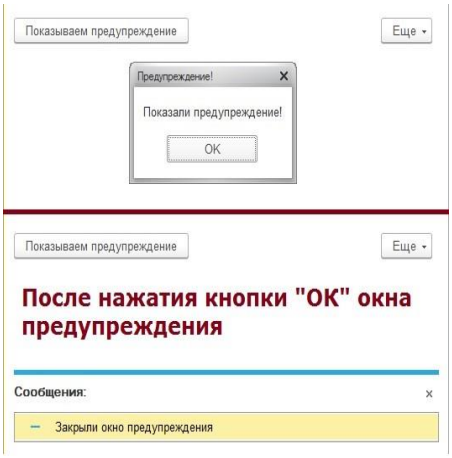


Рис. 13.2. Вывод сообщения после нажатия кнопки «ОК»

6) В процедуру обработчика ожидания ПослеЗакрытияПредупреждения можно также передать какой-нибудь параметр

```
&НаКлиенте
Процедура ПоказываемПредупреждение(Команда)
     ПараметрВОписание = Новый Структура();
     ПараметрВОписание.Вставить("Сообщение","Мы закрыли предупреждение");

     ОписаниеОповещения = Новый
     ОписаниеОповещения("ПослеЗакрытияПредупреждения",ЭтаФорма,ПараметрВОписание);
     ПоказатьПредупреждение(ОписаниеОповещения,"Показали предупреждение!",,"Предупреждение!");
КонецПроцедуры

&НаКлиенте
Процедура ПослеЗакрытияПредупреждения(Параметры) Экспорт
      Сообщить(Параметры.Сообщение);
КонецПроцедуры
```

Листинг 13.3. Передача параметра в оповещение

7) Проверить, как будет выполняться код с листинга 13.3.

Обратите внимание, что процедура оповещения (в примере это *ПослеЗакрытияПредупреждения*) будет выполняться только после того, как отработает весь код в процедуре, в которой было вызвано предупреждение.

8) Для проверки этого факта доработать код с листинга 13.3: запустить после метода *ПоказатьПредупреждение* небольшой цикл, работу которого вывести в сообщение.

```
&НаКлиенте
Процедура ПоказываемПредупреждение(Команда)
    ПараметрВОписание = Новый Структура();
    ПараметрВОписание.Вставить("Сообщение","Мы закрыли предупреждение");

    ОписаниеОповещения = Новый
    ОписаниеОповещения("ПослеЗакрытияПредупреждения",ЭтаФорма,ПараметрВОписание);
    ПоказатьПредупреждение(ОписаниеОповещения,"Показали предупреждение!",,"Предупреждение!");

    Для н = 0 по 4 Цикл
        НомерПоПорядку = (н+1);
        Сообщить("Номер по порядку " + НомерПоПорядку);
    КонецЦикла;
КонецПроцедуры

&НаКлиенте
Процедура ПослеЗакрытияПредупреждения(Параметры) Экспорт
     Сообщить(Параметры.Сообщение);
КонецПроцедуры
```

Листинг 13.4. Вывод цикла после процедуры «ПоказатьПредупреждение»

9) Проверить, как отработает этот код.

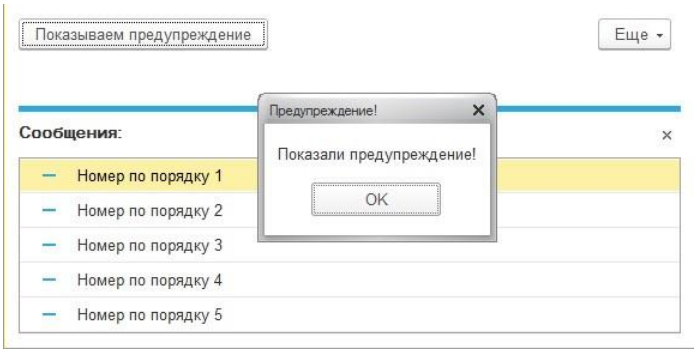


Рис. 13.3. Вывод массива до нажатия кнопки «ОК»

В данном случае не нужно ждать нажатия кнопки «ОК», чтобы выполнить код после метода **ПоказатьПредупреждение**, код будет выполняться дальше в независимости от действий пользователя, и только после того, как он нажмет кнопку «ОК», выйдет соответствующее сообщение (см. рис. 13.4).

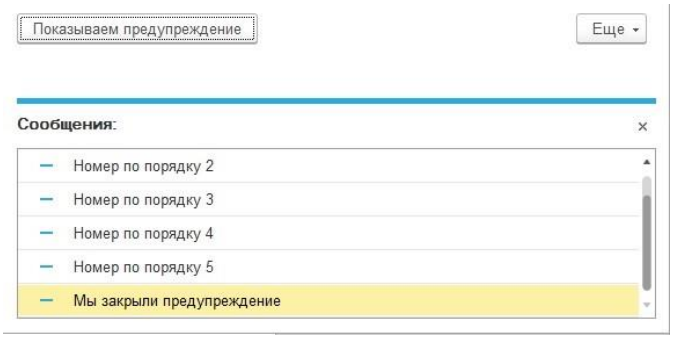


Рис. 13.4. Вывод сообщения после нажатия кнопки «ОК»

Основное отличие всех ***немодальных методов*** от модальных в том, что ввод-вывод информации осуществляется асинхронно. Пока пользователь решает, нажать или не нажать кнопку «ОК», как ответить на вопрос, или какое число ввести, идет выполнение кода программы.

Для этого в метод передается название процедуры, которая будет обработчиком ожидания и сработает, когда пользователь нажмет кнопку «ОК».

В то же время не стоит путать асинхронность с параллельностью. Код, который сработает в процедуре обработчика ожидания, не будет выполняться параллельно основному коду. Он сработает после выполнения этого кода.

2. ПоказатьВопрос

Один из основных методов интерактивной работы с пользователем - метод **ПоказатьВопрос**. При помощи этого метода можно вывести окно сообщения, в котором могут быть кнопки «Да», «Нет», «Ок» и т.д.

Синтаксис метода:

ПоказатьВопрос(<ОписаниеОповещенияОЗавершении>, <ТекстВопроса>, <Кнопки>, <Таймаут>, <КнопкаПоУмолчанию>, <Заголовок>, <КнопкаТаймаута>)

Где:

ОписаниеОповещенияОЗавершении – переменная, которая имеет тип «ОписаниеОповещения», и содержит процедуру, которая сработает после нажатия на ту или иную кнопку;

ТекстВопроса – текст вопроса;

Кнопки – или системное перечисление *РежимДиалогаВопрос*, которое определяет, какие кнопки должны быть в окне вопроса, или список значений, где перечислены все кнопки окна.

Таймаут – интервал времени в секундах, в течение которого будет отображаться окно предупреждения. Если параметр не указан, то время не ограничено;

КнопкаПоУмолчанию – определяет кнопку по умолчанию;

Заголовок – заголовок окна предупреждения;

КнопкаТаймаута – определяет кнопку, на которой будет отображаться таймаут.

В случае метода **ПоказатьВопрос**, параметр *ОписаниеОповещенияОЗавершении* – обязательный, а в процедуре, которая описана в оповещении, должно быть два параметра: Результат и Параметры.

Пример 2.

1) Добавить новую команду **Показать вопрос** и поместить её на форму
2) В обработчике этой команды использовать самый простой вариант процедуры ПоказатьВопрос - выводить окно с кнопками «Да» - «Нет» и обрабатывать нажатие на эти кнопки.

```
&НаКлиенте
Процедура ПоказываемВопрос(Команда)
      ОписаниеОповещения = Новый ОписаниеОповещения("ПослеЗакрытияВопроса",ЭтаФорма);
      ПоказатьВопрос(ОписаниеОповещения,"Нужно ответить данет",РежимДиалогаВопрос.ДаНет);
КонецПроцедуры

&НаКлиенте
Процедура ПослеЗакрытияВопроса(Результат, Параметр) Экспорт
      Если Результат = КодВозвратаДиалога.Да Тогда
            Сообщить("Ответили да!");
       ИначеЕсли Результат = КодВозвратаДиалога.Нет Тогда
            Сообщить("Ответили нет!");
    КонецЕсли;
КонецПроцедуры
```

Листинг 13.5. Использование метода «ПоказатьВопрос»

3) Проверить работу команды

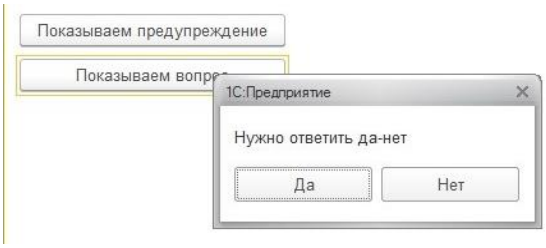


Рис. 13.5. Окно вопроса

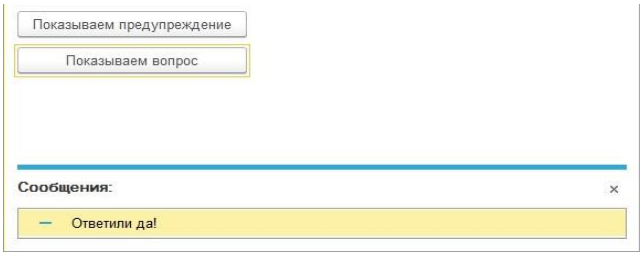


Рис. 13.6. Вывод сообщения при нажатии на кнопку «Да»

В процедуре *ПоказываемВопрос* был создан объект *ОписаниеОповещения*, в котором указана экспортная процедура *ПослеЗакрытияВопроса*, имеющая два параметра:

- первый – результат выполнения вопроса, в данном случае этот параметр будет иметь одно из значений системного перечисления «КодВозваратДиалога», в зависимости от того, какую кнопку нажал пользователь;
- второй – это какой-либо параметр, который можно указать в описании оповещения.

3. ПоказатьВводЧисла

Есть ряд методов, при помощи которых пользователь может интерактивно вводить какие-либо значения в программу. Это методы ПоказатьВводЧисла, ПоказатьВводСтроки, ПоказатьВводДаты и ПоказатьВводЗначения.

Рассмотрим принципы работы этих методов на примере ПоказатьВводЧисла.

Данный метод имеет следующий синтаксис:

ПоказатьВводЧисла(<ОписаниеОповещенияОЗавершении>, <Число>, <Подсказка>, <Длина>, <Точность>)

Где

ОписаниеОповещенияОЗавершении – данный параметр должен иметь тип *ОписаниеОповещения* и указывать процедуру, которая будет вызвана после ввода числа. Данная процедура должна иметь два параметра:

- Первый параметр - это значение числа, которое было введено.
- Второй параметр - это некоторый дополнительный параметр, который будет указан в описании оповещения.

Число – значение числа, которое будет указано в диалоге (по умолчанию).

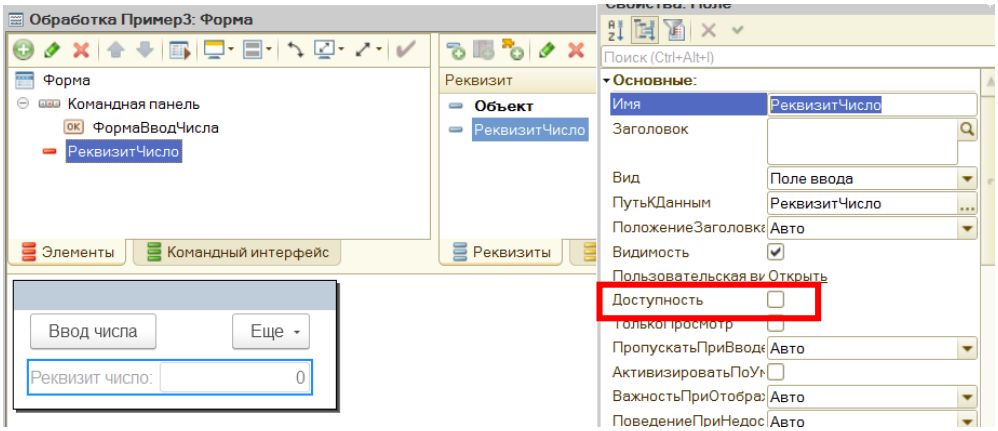
Подсказка - текст заголовка.

Длина - длина вводимого числа, включая дробную часть.

Точность – количество знаков после запятой.

Пример 3.

1) На форме создать реквизит *РеквизитЧисло* с типом «Число» (длина 10, точность 1), разместить его на форме.
2) Закрыть доступ к реквизиту на форме (свойство *Доступность*).



3) Создать команду *ВводЧисла*, в клиентском обработчике которой будет вызываться метод *ПоказатьВводЧисла*

```
&НаКлиенте
Процедура ВводЧисла(Команда)
    Оповещение = Новый ОписаниеОповещения("ПослеВводаЧисла",ЭтаФорма);
    ПоказатьВводЧисла(Оповещение,0,"Введите число",10,1);
КонецПроцедуры

Процедура ПослеВводаЧисла(ВведеноеЧисло, Параметры) Экспорт
    Если ВведеноеЧисло = Неопределено Тогда
      Возврат
    КонецЕсли;

    РеквизитЧисло = ВведеноеЧисло;
КонецПроцедуры
```

Листинг 13.6. Обработчик ввода числа

4) Проверить работу команды Ввод числа
5) Самостоятельно добавьте соответствующие реквизиты и команды для ввода строки, даты и значения.