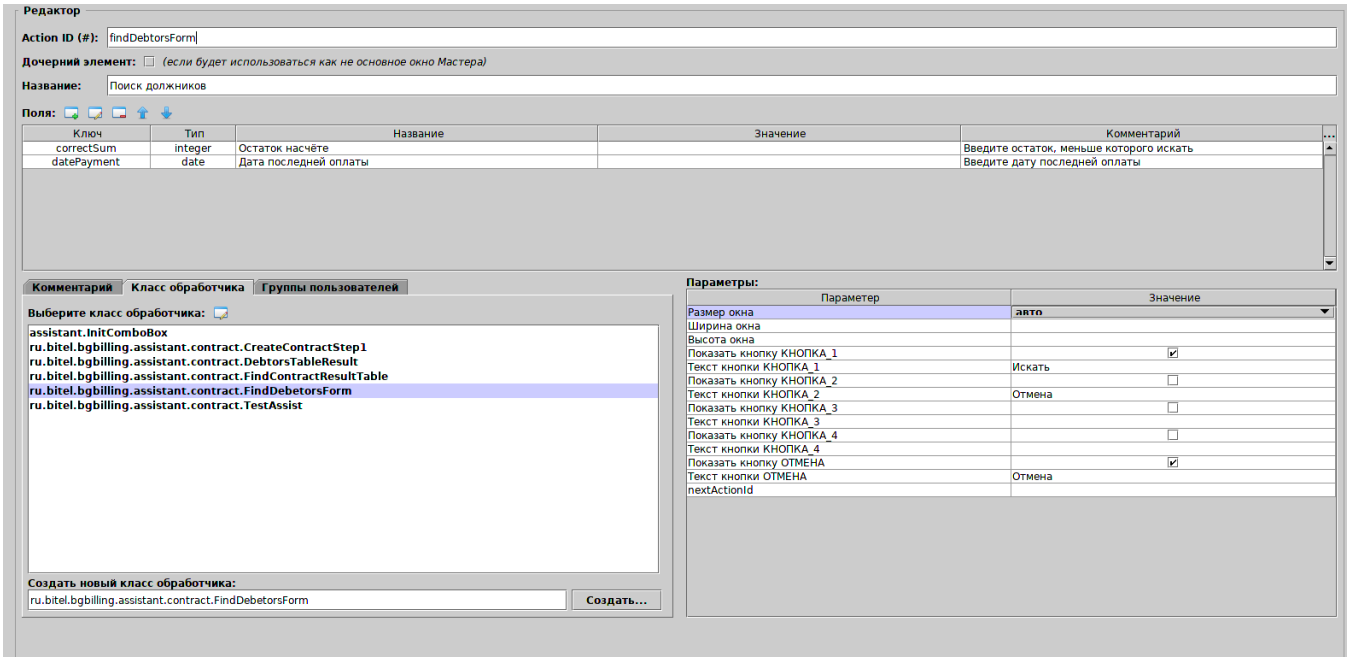
Создание таблицы в мастере

Если таблица содержит колонку с названием "contractId"Результаты действий мастера можно вывести в таблице. Для этого нужно добавить элемент "таблица" в редакторе мастера. В данном примере будет показан весь путь создания небольшого "мастера".

Допустим, что нужен мастер, который отображает в таблице список абонентов с отрицательным балансом, и пополнение счёта от которых не было более 2 месяцев

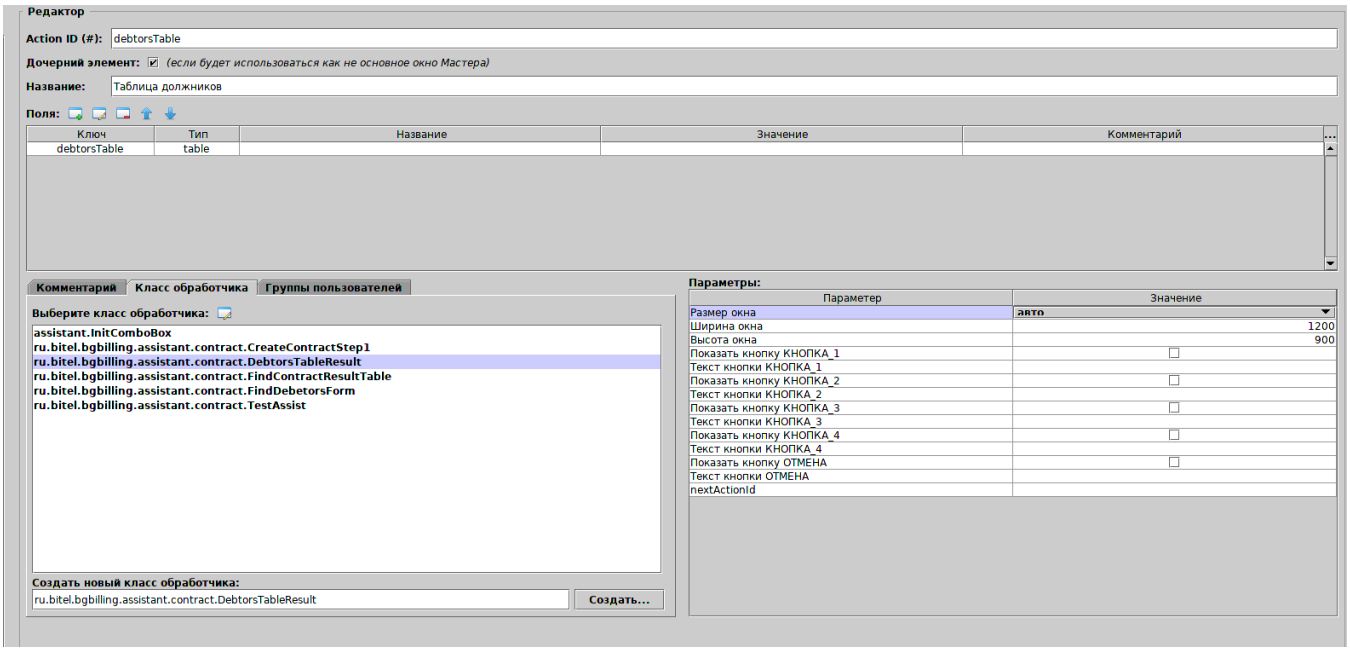


Пример обработчика окна с вводом параметров

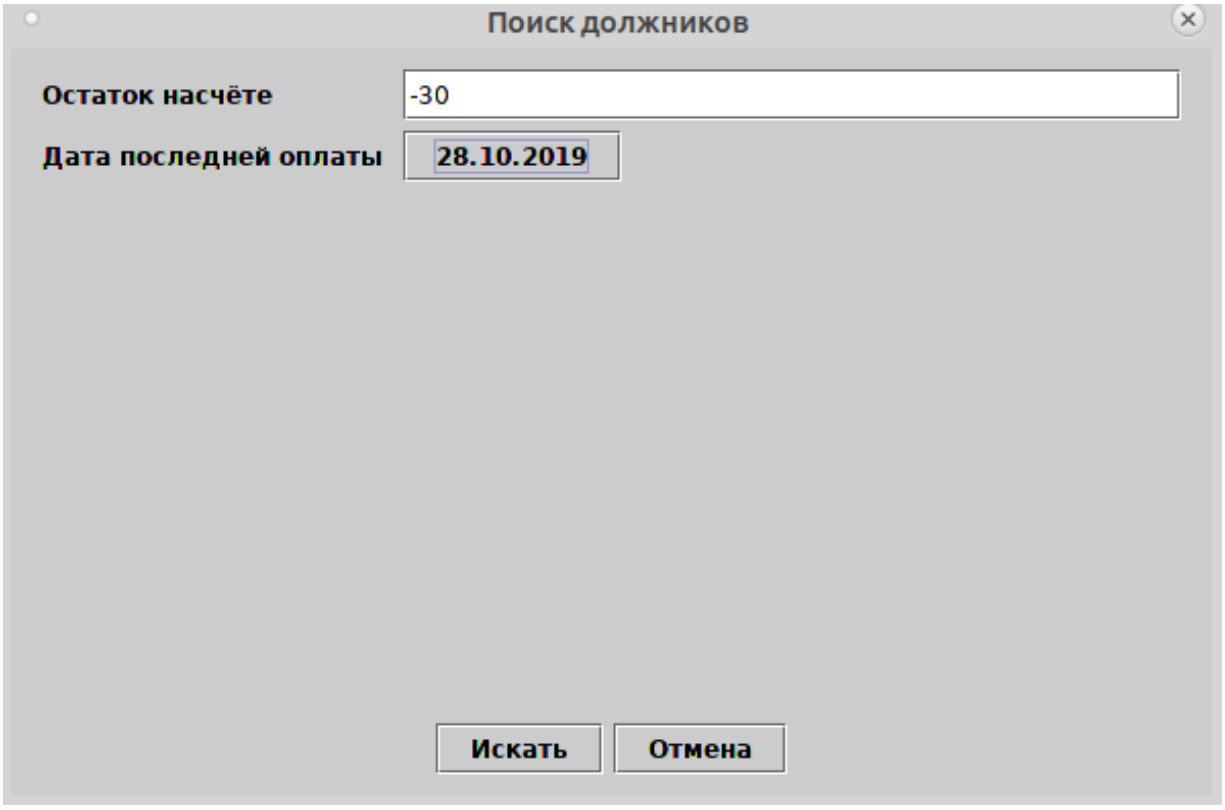
```
public class FindDebetorsForm
    extends AssistantActionBase
{
    @Override
    public AssistantResponse doAction( AssistantRequest assistantRequest )
    throws BGException, BGMessageException
    {
        assistantResponse.setNextActionId( "debtorsTable" ); // Action ID  ,
        return super.doAction( assistantRequest );
    }

    @Override
    public AssistantResponse showAction( AssistantRequest assistantRequest )
    throws BGException, BGMessageException
    {
        this.assistantRequest = assistantRequest;
        return assistantResponse;
    }

    @Override
    public void doButton1()
    {
       super.doButton1();
    }

    @Override
    public void doButton2()
    {
        try
        {
            doAction( assistantRequest );
        }
        catch( BGException e )
        {
            e.printStackTrace();
        }
    }
}
```

Создаём окно с таблицей, указывая **Action ID: debtorsTable**

Добавим элемент "таблица" с ключем: debtorsTable (ключ может быть любой. Если таблиц несколько, то у каждой должен быть свой уникальный ключ)



При запуске мастера "Поиск должников", появится такое окно для задания условий.



В обработчике мы можем получить введённые в мастере данные и произвести необходимые операции с ними. В данном примере показано, как можно получить данные и пример некоторых действий.

Для того, чтобы передать список собственных классов в механизм создания таблицы, нужно передать полный путь до созданного класса, также нужно преобразовать список в формат JSON,

воспользовавшись методом **mapperWrite(List<? extends FilterTableModel> data)** из класса **ru.bitel.bgbilling.plugins.assistant.server.AssistantUtils**

Пример обработчика окна с таблицей.

```
public class DebtorsTableResult
    extends AssistantActionBase
{
    @Override
    public AssistantResponse showAction( AssistantRequest assistantRequest1 )
    throws BGException, BGMessageException
    {
        this.assistantRequest = assistantRequest1;

                //
        JSONObject data = assistantRequest.getData().getJSONObject( "fields" );
        int balanceLimit = Integer.parseInt( data.getString( "correctSum" ) );
        LocalDate searchDate = LocalDate.parse( data.getString( "datePayment" ) );

                //       ,    JSON
        String result = getDataOfTable( balanceLimit, searchDate );
        assistantResponse.addFieldData( "debtorsTable", new AssistantKeyValue[]{ new AssistantKeyValue( "ru.
bitel.bgbilling.assistant.contract.DebtorTableFilter", result )} );

        return super.showAction( assistantRequest );
    }

    private String getDataOfTable( int balanceLimit, LocalDate startSearchDate )
    {
        List<DebtorTableFilter> debtors = new ArrayList<>();
        Connection con = ServerContext.get( ServerContext.class ).getConnection();
        String query = "SELECT c.id, c.title, cp.dt FROM contract AS c "
                     + "LEFT JOIN contract_payment AS cp ON cp.cid=c.id WHERE cp.dt > ? GROUP BY cid ORDER BY
cp.id DESC";

        try( PreparedStatement ps = con.prepareStatement( query ) )
        {
            ps.setDate( 1, Date.valueOf( startSearchDate ) );
            ResultSet rs = ps.executeQuery();

            while( rs.next() )
            {
                DebtorTableFilter debtor  = new DebtorTableFilter();
                debtor.setContractId( rs.getInt( "c.id" ) );
                debtor.setContractTitle( rs.getString( "c.title" ) );
                LocalDate localDate = rs.getDate( "cp.dt" ).toLocalDate();
                debtor.setLastPaymentDate( localDate.toString() );

                debtors.add( debtor );
            }
        }
        catch( SQLException e )
        {
            e.printStackTrace();
        }

        //   ,
        BalanceUtils balanceUtils = new BalanceUtils( con );
        Iterator<DebtorTableFilter> iterator = debtors.iterator();
        while( iterator.hasNext() )
        {
            DebtorTableFilter deb = iterator.next();
            BigDecimal balance = balanceUtils.getBalance( new java.util.Date(  ), deb.getContractId() );
            if( balance.compareTo( new BigDecimal( balanceLimit ) ) >= 0 )
            {
                iterator.remove();
            }
            else
            {
                deb.setBalance( balance );
            }
        }

        //
        for( DebtorTableFilter deb : debtors )
        {
```

```
            String query1 = "SELECT mail.email, phone.value FROM contract AS c LEFT JOIN
contract_parameter_type_3 AS mail ON mail.cid=c.id LEFT JOIN contract_parameter_type_phone AS phone ON phone.
cid=c.id WHERE c.id=?";
            try( PreparedStatement ps = con.prepareStatement(query1) )
            {
                ps.setInt(1, deb.getContractId() );

                ResultSet rs = ps.executeQuery();

                while( rs.next() )
                {
                    deb.setEmail( rs.getString( "mail.email" ) );
                    deb.setTelephone( rs.getString( "phone.value" ) );
                }
            }
            catch( SQLException ex )
            {
                e.printStackTrace();
            }
        }

                //       List    JSON-
        return AssistantUtils.mapperWrite( debtors );
    }
}
```

Для того, чтобы данные корректно отобразились в таблице, необходимо в дин.коде создать собственный класс, который должен быть наследником абстрактного класса **ru.bitel.bgbilling.plugins.assistant.common.bean.FilterTableModel.**

Пример пользовательского класса, который будет отображён в таблице

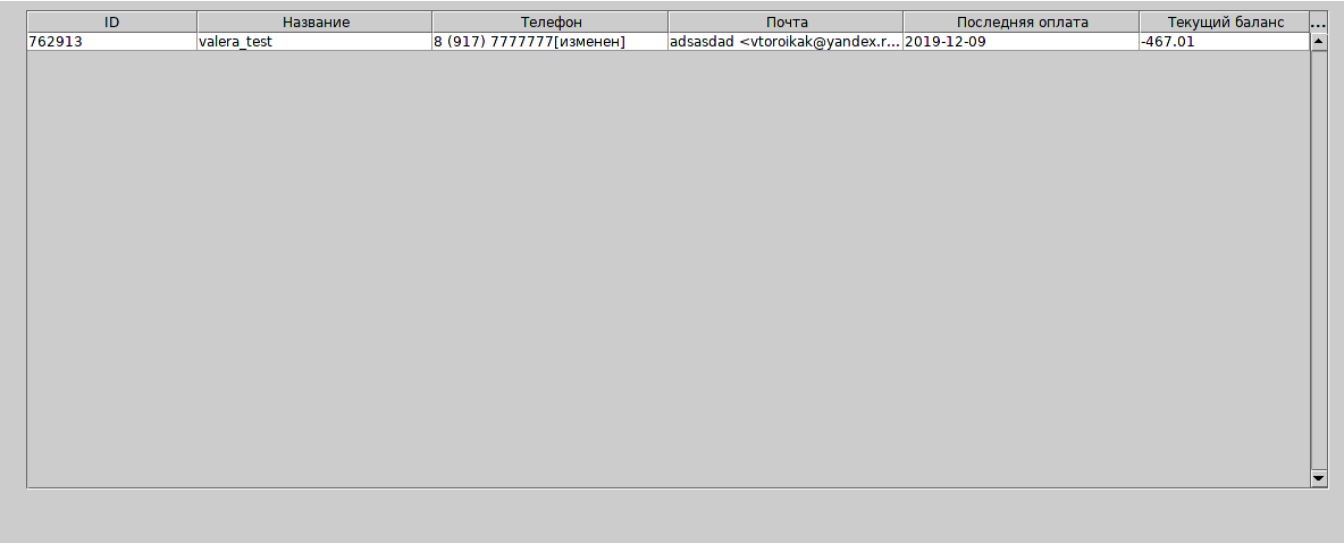
```
@JsonAutoDetect
public class DebtorTableFilter
    extends FilterTableModel
{
    private int contractId;
    private String contractTitle;
    private String telephone;
    private String email;
    private String lastPaymentDate;
    private BigDecimal balance;

    @Override
    public String getCorrectTitleColumn( int columnId )
    {
        switch( columnId )
        {
            case 0: return "ID ";
            case 1: return "";
            case 2: return "";
            case 3: return "";
            case 4: return " ";
            case 5: return " ";
            default: return "< >";
        }
    }

    @Override
    public String getCorrectData( int columnId )
    {
        return null;
    }

    public int getContractId()
    {
        return contractId;
    }

    public void setContractId( int contractId )
```

```
    {
        this.contractId = contractId;
    }

    public String getContractTitle()
    {
        return contractTitle;
    }

    public void setContractTitle( String contractTitle )
    {
        this.contractTitle = contractTitle;
    }

    public String getTelephone()
    {
        return telephone;
    }

    public void setTelephone( String telephone )
    {
        this.telephone = telephone;
    }

    public String getEmail()
    {
        return email;
    }

    public void setEmail( String email )
    {
        this.email = email;
    }

    public String getLastPaymentDate()
    {
        return lastPaymentDate;
    }

    public void setLastPaymentDate( String lastPaymentDate )
    {
        this.lastPaymentDate = lastPaymentDate;
    }

    public BigDecimal getBalance()
    {
        return balance;
    }

    public void setBalance( BigDecimal balance )
    {
        this.balance = balance;
    }
}
```

При необходимости, можно реализовать абстрактный метод **getCorrectData()** и передать в таблицу данные для определённого столбца. Например, есть необходимость показать в таблицы текущий статус для договоров, с помощью этого метода, можно подменить число статуса на более понятный "Активен". Если нет необходимости в "подмене" данных, метод должен возвращать null.

Второй метод **getCorrectTitleColumn()** из родительского FilterTableModel должен возвращать корректное название для колонок таблицы.

Итоговый результат работы мастера показан ниже



Если таблица содержит колонку с названием "contractId", "cid" , "idContract" или "ID Договора", то при клике на строке, будет выполняться поиск договора с таким id в БД, если договор будет найден, он будет открыт.

Есть возможность передать выбранную строку в таблице в обработчик следующего Мастера. Для этого нужно выбрать в таблице строку, при клике правой кнопкой мыши появится меню с пунктом "Сохранить строку для обработчика следующего Мастера", при клике данные буду сохранены и будет возможность получить их в следующем обработчике. Пример получения показан ниже

Пример получения выбранных в таблице данных

```
public class TestViewSelectedRowData
    extends AssistantActionBase
{
    @Override
    public AssistantResponse showAction( AssistantRequest assistantRequest1 )
    throws BGException, BGMessageException
    {
        this.assistantRequest = assistantRequest1;

                //         data
        JSONObject data = assistantRequest.getData().getJSONObject( "fields" );
        if( data.has( "debtorsTable" ) )
        {
                        //        Json
            JSONObject selected = data.getJSONObject( "debtorsTable" );

                        //  -     .
            String contractTitle = (String)selected.get( "contractTitle" );
            BigDecimal balance = new BigDecimal( selected.get( "balance" ).toString() );
            String email = (String)selected.get( "email" );
        }

        return super.showAction( assistantRequest );
    }
}
```

Выбранная в таблице строка будет сохранена под ключом самой таблицы, в данном примере ключом является строка "debtorsTable". Конкретные данные для выбранной в таблице строки сохраняются под ключом поля объекта, который был представлен в таблице. В данном случае в объекте DebtorTableFilter есть поля: contractTitle, balance и email. Используя названия этих полей как ключи, можно получить значения.