

# FIREBIRD CONF 2023

Написание UDR на языках C++ и Pascal

Организатор  
конференции



РЕДСОФТ



# Написание UDR на языках C++ и Pascal

Симонов Денис



# Отличия UDR от Legacy UDF

---

## UDR

- Функции, процедуры, триггеры
- Строгий контроль типов входных и выходных параметров. Простая обработка NULL.
- Есть обработка исключительных ситуаций
- Доступ к контексту текущего подключения и транзакции
- Можно группировать процедуры и функции в PSQL пакетах
- Могут быть написаны на любом языке программирования, если есть соответствующий плагин External Engine. Например, существуют плагины для написания на Java, .NET. Для C++ и Pascal используется стандартный плагин `udr_engine`.

## Legacy UDF

- Только функции
- Слабый тип входных и выходных параметров. Несколько вариантов обработки NULL.
- Возврат ошибок через специальные значения
- Более быстрые за счёт упрощенного контроля типов входных и выходных параметров
- Только низкоуровневые языки программирования Pascal, C, C++.

# Синтаксис объявления UDR

```
{CREATE [OR ALTER] | RECREATE} FUNCTION funcname [(<inparam> [, <inparam> ...])]  
RETURNS <type> [COLLATE collation] [DETERMINISTIC]  
EXTERNAL NAME <extname> ENGINE <engine>  
[AS <extbody>]
```

```
{CREATE [OR ALTER] | RECREATE} PROCEDURE procname [(<inparam> [, <inparam> ...])]  
RETURNS (<outparam> [<outparam> ...])  
EXTERNAL NAME <extname> ENGINE <engine>  
[AS <extbody>]
```

```
{CREATE [OR ALTER] | RECREATE} TRIGGER trigname  
{<relation_trigger_legacy> | <relation_trigger_sql2003> | <database_trigger> | <ddl_trigger> }  
EXTERNAL NAME <extname> ENGINE <engine>  
[AS <extbody>]
```

**<extname>** ::= '**<module name>!<routine name>[!<misc info>]**'

**<extbody>** ::= произвольный строковый литерал.

# Пример внешних функции

---

```
CREATE FUNCTION sum_args (n1 INTEGER, n2 INTEGER, n3 INTEGER)  
RETURNS INTEGER  
EXTERNAL NAME 'udrcpp_example!sum_args' ENGINE UDR;
```

```
CREATE FUNCTION regex_replace (regex VARCHAR(60), str VARCHAR (60), replacement VARCHAR (60))  
RETURNS varchar(60)  
EXTERNAL NAME 'org.firebirdsql.fbjava.examples.fbjava_example.FbRegex.replace(String, String, String)'  
ENGINE java;
```

# Пример внешних процедур

---

```
CREATE PROCEDURE gen_rows_pascal (start_n INTEGER NOT NULL, end_n INTEGER NOT NULL)  
RETURNS (result INTEGER)  
EXTERNAL NAME 'pascaludr!gen_rows' ENGINE udr;
```

```
CREATE OR ALTER PROCEDURE write_log (message VARCHAR(100))  
EXTERNAL NAME 'pascaludr!write_log' ENGINE udr;
```

```
CREATE OR ALTER PROCEDURE employee_pgsql (  
    dummy INTEGER = 1  
)  
RETURNS(id TYPE OF COLUMN employee.id, name TYPE OF COLUMN employee.name)  
EXTERNAL NAME 'org.firebirdsql.fbjava.examples.fbjava_example.FbJdbc  
.executeQuery()!jdbc:postgresql:employee|postgres|postgres'  
ENGINE java  
AS Q'{select * from employee}';
```

# Пример внешних триггеров

---

```
CREATE TABLE persons (  
  id INTEGER NOT NULL,  
  name VARCHAR(60) NOT NULL,  
  address VARCHAR(60),  
  info BLOB SUB_TYPE TEXT  
);
```

```
CREATE TABLE replicate_config (name VARCHAR(63) NOT NULL, data_source VARVHAR(255) NOT NULL );
```

```
INSERT INTO replicate_config (name, data_source) VALUES ('ds1', 'c:\temp\slave.fdb');
```

```
CREATE TRIGGER persons_replicate  
AFTER INSERT ON persons  
EXTERNAL NAME 'udrcpp_example!replicate!ds1' ENGINE udr;
```

# Использование UDR в пакетах

---

```
CREATE OR ALTER PACKAGE REGEXP  
AS  
BEGIN
```

```
PROCEDURE preg_match( APattern VARCHAR(8191), ASubject VARCHAR(8191))  
RETURNS (Matches VARCHAR(8191));
```

```
FUNCTION preg_replace(APattern VARCHAR(8191), AReplacement VARCHAR(8191), ASubject VARCHAR(8191))  
RETURNS VARCHAR(8191);
```

```
PROCEDURE preg_split( APattern VARCHAR(8191), ASubject VARCHAR(8191))  
RETURNS (Lines VARCHAR(8191));
```

```
FUNCTION preg_quote( AStr VARCHAR(8191), ADelimiter CHAR(10) DEFAULT NULL)  
RETURNS VARCHAR(8191);
```

```
END
```



# Использование UDR в пакетах

---

```
RECREATE PACKAGE BODY REGEXP
AS
BEGIN
```

```
PROCEDURE preg_match( APattern VARCHAR(8192), ASubject VARCHAR(8192))
RETURNS (Matches VARCHAR(8192))
EXTERNAL NAME 'PCRE!preg_match' ENGINE UDR;
```

```
FUNCTION preg_replace(APattern VARCHAR(8192), AReplacement VARCHAR(8192), ASubject VARCHAR(8192))
RETURNS VARCHAR(8192)
EXTERNAL NAME 'PCRE!preg_replace' ENGINE UDR;
```

```
...
END
```

# Точка входа на языке Pascal

---

```
library MyUdr;  
{ $IFDEF FPC }  
  { $MODE DELPHI } { $H+ }  
{ $ENDIF }  
  
uses  
{ $IFDEF unix }  
  cthreads, // the c memory manager is on some systems much faster for multi-threading  
  cmem,  
{ $ENDIF }  
  UdrInit in 'UdrInit.pas',  
  SumArgsFunc in 'SumArgsFunc.pas';  
  
exports firebird_udr_plugin;  
  
end.
```

# Точка входа на языке C++

---

```
#define FB_UDR_STATUS_TYPE ::Firebird::ThrowStatusWrapper
```

```
#include "ibase.h"
```

```
#include "firebird/UdrCppEngine.h"
```

```
using namespace Firebird;
```

```
...
```

```
FB_UDR_IMPLEMENT_ENTRY_POINT
```

В качестве альтернативы можно использовать CheckStatusWrapper

# Регистрация фабрик в Pascal

---

```
unit UdrInit;
```

```
interface
```

```
uses Firebird;
```

```
function firebird_uds_plugin(AStatus: IStatus; AUnloadFlagLocal: BooleanPtr; AUdrPlugin: IUdrPlugin): BooleanPtr; cdecl;
```

```
implementation
```

```
uses SumArgsFunc;
```

```
var myUnloadFlag: Boolean; theirUnloadFlag: BooleanPtr;
```

```
function firebird_uds_plugin (AStatus: IStatus; AUnloadFlagLocal: BooleanPtr; AUdrPlugin: IUdrPlugin): BooleanPtr; cdecl;
```

```
begin
```

```
  AUdrPlugin.registerFunction(AStatus, 'sum_args', TSumArgsFunctionFactory.Create());
```

```
  AUdrPlugin.registerProcedure(AStatus, 'sum_args_proc', TSumArgsProcedureFactory.Create
```

```
  AUdrPlugin.registerTrigger(AStatus, 'test_trigger', TMyTriggerFactory.Create());
```

```
  theirUnloadFlag := AUnloadFlagLocal;
```

```
  Result := @myUnloadFlag;
```

```
end;
```

```
initialization
```

```
  myUnloadFlag := false;
```

```
finalization
```

```
  if ((theirUnloadFlag <> nil) and not myUnloadFlag) then theirUnloadFlag^ := true;
```

```
end.
```

# Реализация фабрики функции на Pascal

---

```
unit SumArgsFunc;
```

```
interface
```

```
uses Firebird;
```

```
type
```

```
TSumArgsInMsg = record
```

```
  n1: Integer; n1Null: WordBool;
```

```
  n2: Integer; n2Null: WordBool;
```

```
  n3: Integer; n3Null: WordBool;
```

```
end;
```

```
PSumArgsInMsg = ^TSumArgsInMsg;
```

```
TSumArgsOutMsg = record
```

```
  result: Integer; resultNull: WordBool;
```

```
end;
```

```
PSumArgsOutMsg = ^TSumArgsOutMsg;
```

```
TSumArgsFunctionFactory = class(IUdrFunctionFactoryImpl)
```

```
  procedure dispose(); override;
```

```
  procedure setup(AStatus: IStatus; AContext: IExternalContext;
```

```
    AMetadata: IRoutineMetadata; AInBuilder: IMetadataBuilder;
```

```
    AOutBuilder: IMetadataBuilder); override;
```

```
  function newItem(AStatus: IStatus; AContext: IExternalContext;
```

```
    AMetadata: IRoutineMetadata): IExternalFunction; override;
```

```
end;
```

```
...
```

# Реализация фабрики функции на Pascal

---

## implementation

```
{ TSumArgsFunctionFactory }
```

```
procedure TSumArgsFunctionFactory.dispose;
```

```
begin
```

```
    Destroy;
```

```
end;
```

```
function TSumArgsFunctionFactory.newItem(AStatus: IStatus;
```

```
    AContext: IExternalContext; AMetadata: IRoutineMetadata): IExternalFunction;
```

```
begin
```

```
    Result := TSumArgsFunction.Create();
```

```
end;
```

```
procedure TSumArgsFunctionFactory.setup(AStatus: IStatus;
```

```
    AContext: IExternalContext; AMetadata: IRoutineMetadata;
```

```
    AInBuilder, AOutBuilder: IMetadataBuilder);
```

```
begin
```

```
end;
```

```
...
```

# Реализация функции на Pascal

---

```
unit SumArgsFunc;
```

```
interface
```

```
uses Firebird;
```

```
type
```

```
...
```

```
TSumArgsInMsg = record
```

```
  n1: Integer; n1Null: WordBool;
```

```
  n2: Integer; n2Null: WordBool;
```

```
  n3: Integer; n3Null: WordBool;
```

```
end;
```

```
PSumArgsInMsg = ^TSumArgsInMsg;
```

```
TSumArgsOutMsg = record
```

```
  result: Integer; resultNull: WordBool;
```

```
end;
```

```
PSumArgsOutMsg = ^TSumArgsOutMsg;
```

```
TSumArgsFunction = class(IExternalFunctionImpl)
```

```
  procedure dispose(); override;
```

```
  procedure getCharSet(AStatus: IStatus; AContext: IExternalContext;
```

```
    AName: PAnsiChar; ANameSize: Cardinal); override;
```

```
  procedure execute(AStatus: IStatus; AContext: IExternalContext;
```

```
    AInMsg: Pointer; AOutMsg: Pointer); override;
```

```
end;
```

# Реализация функции на Pascal

---

## implementation

...

{ TSumArgsFunction }

**procedure** TSumArgsFunction.dispose;

**begin**

  Destroy;

**end;**

**procedure** TSumArgsFunction.execute(AStatus: IStatus; AContext: IExternalContext; AInMsg, AOutMsg: Pointer);

**var**

  xInput: PSumArgsInMsg;

  xOutput: PSumArgsOutMsg;

**begin**

  // преобразовываем указатели на вход и выход к типизированным

  xInput := PSumArgsInMsg(AInMsg);

  xOutput := PSumArgsOutMsg(AOutMsg);

  // если один из аргументов NULL значит и результат NULL

  xOutput^.resultNull := xInput^.n1Null or xInput^.n2Null or xInput^.n3Null;

  xOutput^.result := xInput^.n1 + xInput^.n2 + xInput^.n3;

**end;**

**procedure** TSumArgsFunction.getCharSet(AStatus: IStatus; AContext: IExternalContext; AName: PAnsiChar; ANameSize: Cardinal);

**begin**

**end;**

**end.**



# Реализация функции на C++

---

```
#include "UdrCppExample.h"

using namespace Firebird;

struct TSumArgsInMsg
{
    ISC_LONG n1; ISC_SHORT n1Null;
    ISC_LONG n2; ISC_SHORT n2Null;
    ISC_LONG n3; ISC_SHORT n3Null;
};

struct TSumArgsOutMsg
{
    ISC_LONG result; ISC_SHORT resultNull;
};

FB_UDR_BEGIN_FUNCTION(sum_args)
    FB_UDR_EXECUTE_FUNCTION
    {
        auto input = reinterpret_cast<TSumArgsInMsg*>(in);
        auto output = reinterpret_cast<TSumArgsOutMsg*>(out);

        output->resultNull = (input->n1Null || input->n2Null || input->n3Null) ? FB_TRUE : FB_FALSE;
        output->result = input->n1 + input->n2 + input->n3;
    }
FB_UDR_END_FUNCTION

FB_UDR_IMPLEMENT_ENTRY_POINT
```

# Реализация фабрики процедуры на Pascal

---

```
unit GenRowsProc;
```

```
interface
```

```
uses
```

```
  Firebird, SysUtils;
```

```
type
```

```
TGenRowsFactory = class(IUdrProcedureFactoryImpl)
```

```
  procedure dispose(); override;
```

```
  procedure setup(AStatus: IStatus; AContext: IExternalContext;  
    AMetadata: IRoutineMetadata; AInBuilder: IMetadataBuilder;  
    AOutBuilder: IMetadataBuilder); override;
```

```
  function newItem(AStatus: IStatus; AContext: IExternalContext;  
    AMetadata: IRoutineMetadata): IExternalProcedure; override;  
end;
```

```
...
```

# Реализация фабрики процедуры на Pascal

---

## implementation

```
{ TGenRowsFactory }
```

```
procedure TGenRowsFactory.dispose;  
begin  
  Destroy;  
end;
```

```
function TGenRowsFactory.newItem(AStatus: IStatus; AContext: IExternalContext;  
  AMetadata: IRoutineMetadata): IExternalProcedure;  
begin  
  Result := TGenRowsProcedure.create;  
end;
```

```
procedure TGenRowsFactory.setup(AStatus: IStatus; AContext: IExternalContext;  
  AMetadata: IRoutineMetadata; AInBuilder, AOutBuilder: IMetadataBuilder);  
begin  
end;
```

```
...
```

# Реализация процедуры на Pascal

---

**interface**

**uses** Firebird, SysUtils;

**type**

TInput = **record**

start: Integer; startNull: WordBool;

finish: Integer; finishNull: WordBool;

**end;**

PInput = ^TInput;

TOutput = **record**

n: Integer; nNull: WordBool;

**end;**

POutput = ^TOutput;

TGenRowsProcedure = **class**(IExternalProcedureImpl)

**public**

**procedure** dispose(); **override;**

**procedure** getCharSet(AStatus: IStatus; AContext: IExternalContext;

AName: PAnsiChar; ANameSize: Cardinal); **override;**

**function** open(AStatus: IStatus; AContext: IExternalContext; AInMsg: Pointer;

AOutMsg: Pointer): IExternalResultSet; **override;**

**end;**

TGenRowsResultSet = **class**(IExternalResultSetImpl)

Input: PInput;

Output: POutput;

**procedure** dispose(); **override;**

**function** fetch(AStatus: IStatus): Boolean; **override;**

**end;**

# Реализация процедуры на Pascal

---

```
function TGenRowsProcedure.open(AStatus: IStatus; AContext: IExternalContext;
  AInMsg, AOutMsg: Pointer): IExternalResultSet;
begin
  Result := TGenRowsResultSet.create;
  with TGenRowsResultSet(Result) do
    begin
      Input := AInMsg;
      Output := AOutMsg;
    end;

  // если один из входных аргументов NULL ничего не возвращаем
  if PInput(AInMsg).startNull or PInput(AInMsg).finishNull then
    begin
      POutput(AOutMsg).nNull := True;
      // намеренно ставим выходной результат таким, чтобы
      // метод TGenRowsResultSet.fetch вернул false
      Output.n := Input.finish;
    exit;
  end;

  // проверки
  if PInput(AInMsg).start > PInput(AInMsg).finish then
    raise Exception.Create('First parameter greater than second parameter.');
```

```
  with TGenRowsResultSet(Result) do
    begin
      // начальное значение
      Output.nNull := False;
      Output.n := Input.start - 1;
    end;
  end;
```

# Реализация процедуры на Pascal

---

```
{ TGenRowsResultSet }
```

```
procedure TGenRowsResultSet.dispose;
```

```
begin
```

```
    Destroy;
```

```
end;
```

```
// Если возвращает True то извлекается очередная запись из набора данных.
```

```
// Если возвращает False то записи в наборе данных закончились
```

```
// новые значения в выходном векторе вычисляются каждый раз
```

```
// при вызове этого метода
```

```
function TGenRowsResultSet.fetch(AStatus: IStatus): Boolean;
```

```
begin
```

```
    Inc(Output.n);
```

```
    Result := (Output.n <= Input.finish);
```

```
end;
```

# Реализация процедуры на C++

---

```
#include "UdrCppExample.h"
```

```
using namespace Firebird;
```

```
struct TInput
```

```
{  
    ISC_LONG start; ISC_SHORT startNull;  
    ISC_LONG finish; ISC_SHORT finishNull;  
};
```

```
struct TOutput
```

```
{  
    ISC_LONG result; ISC_SHORT resultNull;  
};
```

# Реализация процедуры на C++

---

```
FB_UDR_BEGIN_PROCEDURE(gen_rows)
  FB_UDR_EXECUTE_PROCEDURE
  {
    input = reinterpret_cast<TInput*>(in);
    output = reinterpret_cast<TOutput*>(out);

    output->resultNull = (input->startNull || input->finishNull) ? FB_TRUE : FB_FALSE;
    if (output->resultNull) {
      output->result = input->finish;
    } else {
      if (input->start > input->finish) {
        ISC_STATUS statusVector[] = {
          isc_arg_gds, isc_random,
          isc_arg_string, (ISC_STATUS)"First parameter greater then second parameter.",
          isc_arg_end
        };
        throw Firebird::FbException(status, statusVector);
      }
    }
  }

  TInput* input = nullptr;
  TOutput output = nullptr;

  FB_UDR_FETCH_PROCEDURE
  {
    return output->result++ < input->finish;
  }
FB_UDR_END_PROCEDURE
```



# Реализация фабрики триггера на Pascal

---

```
unit TestTrigger;
```

```
interface
```

```
uses
```

```
    Firebird, SysUtils;
```

```
type
```

```
TMyTriggerFactory = class(IUdrTriggerFactoryImpl)
```

```
    procedure dispose(); override;
```

```
    procedure setup(AStatus: IStatus; AContext: IExternalContext;  
        AMetadata: IRoutineMetadata; AFieldsBuilder: IMetadataBuilder); override;
```

```
    function newItem(AStatus: IStatus; AContext: IExternalContext;  
        AMetadata: IRoutineMetadata): IExternalTrigger; override;
```

```
end;
```

# Реализация фабрики триггера на Pascal

---

## implementation

```
{ TMyTriggerFactory }
```

```
procedure TMyTriggerFactory.dispose;  
begin  
    Destroy;  
end;
```

```
function TMyTriggerFactory.newItem(AStatus: IStatus; AContext: IExternalContext;  
    AMetadata: IRoutineMetadata): IExternalTrigger;  
begin  
    Result := TMyTrigger.create;  
end;
```

```
procedure TMyTriggerFactory.setup(AStatus: IStatus; AContext: IExternalContext;  
    AMetadata: IRoutineMetadata; AFieldsBuilder: IMetadataBuilder);  
begin  
  
end;
```

# Реализация триггера на Pascal

---

**interface**

**type**

TFieldsMessage = **record**

  Id: Integer; IdNull: WordBool;

  A: Integer; ANull: WordBool;

  B: Integer; BNull: WordBool;

  Name: **record**

    Length: Word;

    Value: **array** [0 .. 399] **of** AnsiChar;

**end**;

  NameNull: WordBool;

**end**;

PFieldsMessage = ^TFieldsMessage;

TMyTrigger = **class**(IExternalTriggerImpl)

**procedure** dispose(); **override**;

**procedure** getCharSet(AStatus: IStatus; AContext: IExternalContext;

    AName: PAnsiChar; ANameSize: Cardinal); **override**;

**procedure** execute(AStatus: IStatus; AContext: IExternalContext;

    AAction: Cardinal; AOldMsg: Pointer; ANewMsg: Pointer); **override**;

**end**;

# Реализация триггера на Pascal

---

```
procedure TMyTrigger.execute(AStatus: IStatus; AContext: IExternalContext;  
  AAction: Cardinal; AOldMsg, ANewMsg: Pointer);  
var  
  xOld, xNew: PFieldsMessage;  
begin  
  xOld := PFieldsMessage(AOldMsg);  
  xNew := PFieldsMessage(ANewMsg);  
  case AAction of  
    IExternalTrigger.ACTION_INSERT:  
      begin  
        if xNew.BNull and not xNew.ANull then  
          begin  
            xNew.B := xNew.A + 1;  
            xNew.BNull := False;  
          end;  
        end;  
  
    IExternalTrigger.ACTION_UPDATE:  
      begin  
        if xNew.BNull and not xNew.ANull then  
          begin  
            xNew.B := xNew.A + 1;  
            xNew.BNull := False;  
          end;  
        end;  
  
    IExternalTrigger.ACTION_DELETE:  
      begin  
  
        end;  
      end;  
    end;
```

# Отображение типов данных

Тип данных SQL	C/C++	Pascal	Макрос C/C++
BOOLEAN	ISC_BOOLEAN	Boolean, ByteBool	FB_BOOLEAN
SMALLINT	short	Smallint	FB_SMALLINT
INTEGER	ISC_LONG	Integer	FB_INTEGER
BIGINT		Int64	FB_BIGINT
FLOAT	float	Single	FB_FLOAT
DOUBLE PRECISION	double	Double	FB_DOUBLE
CHAR(N)	char[M]	array of AnsiChar[0 .. M -1]	FB_CHAR
VARCHAR(N)	vary<M>	record Length: Word; value: array of AnsiChar[0 .. M -1]; end	FB_INTL_VARCHAR
TIME	ISC_TIME	ISC_TIME	FB_TIME
DATE	ISC_DATE	ISC_DATE	FB_DATE
TIMESTAMP	ISC_TIMESTAMP	ISC_TIMESTAMP	FB_TIMESTAMP
BLOB	ISC_QUAD	ISC_QUAD	FB_BLOB

# Принудительная установка типов на Pascal

---

```
procedure TSumArgsFunctionFactory.setup(AStatus: IStatus;  
    AContext: IExternalContext; AMetadata: IRoutineMetadata;  
    AInBuilder, AOutBuilder: IMetadataBuilder);  
begin  
    // строим сообщение для входных параметров  
    AInBuilder.setType(AStatus, 0, SQL_LONG + 1);  
    AInBuilder.setType(AStatus, 1, SQL_LONG + 1);  
    AInBuilder.setType(AStatus, 2, SQL_LONG + 1);  
    // строим сообщение для выходных параметров  
    AOutBuilder.setType(AStatus, 0, SQL_LONG + 1);  
end;
```

# Принудительная установка типов на C++

---

```
#include "UdrCppExample.h"
```

```
using namespace Firebird;
```

```
FB_UDR_BEGIN_FUNCTION(sum_args)
    FB_UDR_MESSAGE(InMessage,
        (FB_BIGINT, n1)
        (FB_BIGINT, n2)
        (FB_BIGINT, n3)
    );
```

```
FB_UDR_MESSAGE(OutMessage,
    (FB_BIGINT, result)
);
```

```
FB_UDR_EXECUTE_FUNCTION
{
    out->resultNull = (in->n1Null || in->n2Null || in->n3Null) ? FB_TRUE : FB_FALSE;
    out->result = in->n1 + in->n2 + in->n3;
}
```

```
FB_UDR_END_FUNCTION
```

```
FB_UDR_IMPLEMENT_ENTRY_POINT
```

# Работа с BLOB на Pascal

---

```
var
  att: IAttachment;
  trx: ITransaction;
  blob: IBlob;
  buffer: array [0 .. 32767] of AnsiChar;
  l: Integer;
begin
  try
    att := AContext.getAttachment(AStatus);
    trx := AContext.getTransaction(AStatus);
    blob := att.openBlob(AStatus, trx, ABlobId, 0, nil);
    while True do
      case blob.getSegment(AStatus, SizeOf(buffer), @buffer, @l) of
        IStatus.RESULT_OK:
          AStream.WriteBuffer(buffer, l);
        IStatus.RESULT_SEGMENT:
          AStream.WriteBuffer(buffer, l);
      else
        break;
      end;
    AStream.Position := 0;
    blob.close(AStatus);
    blob := nil;
  finally
    if Assigned(blob) then blob.release;
    if Assigned(trx) then trx.release;
    if Assigned(att) then att.release;
  end;
end;
```



# Работа с BLOB на C++

---

```
Firebird::AutoRelease<Firebird::IAttachment> att(context->getAttachment(status));
Firebird::AutoRelease<Firebird::ITransaction> tra(context->getTransaction(status));
Firebird::AutoRelease<Firebird::IBlob> bodyBlob(att->openBlob(status, tra, &in->body, 0, nullptr));
std::vector<char> vBuffer(MAX_SEGMENT_SIZE);
auto buffer = vBuffer.data();
while (true) {
    unsigned int l = 0;
    switch (bodyBlob->getSegment(status, MAX_SEGMENT_SIZE, buffer, &l))
    {
        case Firebird::IStatus::RESULT_OK:
        case Firebird::IStatus::RESULT_SEGMENT:
            requestBody.write(buffer, l);
            continue;
        default:
            bodyBlob->close(status);
            bodyBlob.release();
            break;
    }
}
// set beginning of stream
requestBody.seekg(0, std::ios::beg);
```

# Контекст соединения и транзакции Pascal

---

```
var
  att: IAttachment;
  trx: ITransaction;
  stmt: IStatement;
begin
  try
    att := AContext.getAttachment(AStatus);
    trx := AContext.getTransaction(AStatus);
    stmt := att.prepare(status, tra,
      0, SQL_APPEND_LOG,
      3, IStatement.PREPARE_PREFETCH_METADATA
    );
    stmt.execute(status, tra,
      input.getMetadata(),
      input.getData(),
      nil, nil
    );
    stmt.free(status);
  finally
    if Assigned(stmt) then stmt.release;
    if Assigned(trx) then trx.release;
    if Assigned(att) then att.release;
  end;
end;
```

# Контекст соединения и транзакции C++

---

```
Firebird::AutoRelease<Firebird::IAttachment> att(context->getAttachment(status));
Firebird::AutoRelease<Firebird::ITransaction> tra(context->getTransaction(status));
Firebird::AutoRelease<Firebird::IStatement> stmt(
    att->prepare(status, tra,
        0, SQL_APPEND_LOG,
        3,
        IStatement::PREPARE_PREFETCH_METADATA
    )
);
stmt->execute(status, tra,
    input.getMetadata(),
    input.getData(),
    nullptr, nullptr
);
stmt->free(status);
stmt.release();
```

# Метод getCharset

---

```
procedure TJsonFunction.getCharSet(AStatus: IStatus; AContext: IExternalContext;  
  AName: PAnsiChar; ANameSize: Cardinal);  
begin  
  FillChar(AName, ANameSize, #0);  
  StrCopy(AName, 'UTF8');  
end;
```

```
FB_UDR_BEGIN_PROCEDURE(ftsLogByDdKey)
```

```
...
```

```
void getCharSet(ThrowStatusWrapper* status, IExternalContext* context,  
  char* name, unsigned nameSize)  
{  
  memset(name, 0, nameSize);  
  
  const std::string charset = "UTF8";  
  charset.copy(name, charset.length());  
}
```

```
FB_UDR_EXECUTE_PROCEDURE
```

```
{  
}
```

```
FB_UDR_END_PROCEDURE
```

# Методы освобождающие интерфейс

---

- IAttachment::detach
- IAttachment::dropDatabase
- ITransaction::commit
- ITransaction::rollback
- IStatement::free
- IResultSet::close
- IBlob::cancel
- IBlob::close
- IService::detach
- IEvents::cancel

# Спасибо за внимание!

---

Вопросы

