

**И. Е. Ермаков**

**Практические аспекты разработки систем на Компонентном Паскале**

Демонстрационный материал к докладу в День Оберона - 2017

Видео докладов: <https://www.youtube.com/playlist?list=PLoKr-Vv5yq7Hdxq1edBtyxvkTehP4t09>

DEFINITION KrcTasks;

IMPORT KrlLife, KrlTokens, KrlEvents;

PROCEDURE **Begin** (OUT id: LONGINT);  
PROCEDURE **Done** (id: LONGINT): BOOLEAN;  
PROCEDURE **I** (): Context;  
PROCEDURE **Stop** (VAR id: LONGINT);

TYPE

**Context** = POINTER TO ABSTRACT RECORD (KrlLife.Object)

(c: Context) **Id** (): LONGINT, NEW, ABSTRACT;

(c: Context) **SlotMs** (): LONGINT, NEW, ABSTRACT;

(c: Context) **Pass**, NEW, ABSTRACT;

(c: Context) **IdleIf** (cond: BOOLEAN), NEW, ABSTRACT;

(c: Context) **Stop**, NEW, ABSTRACT;

(c: Context) **SleepMs** (ms: INTEGER): Context, NEW, ABSTRACT;

(c: Context) **SleepSec** (s: INTEGER): Context, NEW, ABSTRACT;

(c: Context) **Await** (obj: ANYPTR; event: KrlTokens.Token; sp: KrlEvents.Space): Context, NEW, ABSTRACT;

(c: Context) **AwaitEvents** (IN objs: ARRAY OF ANYPTR; IN events: ARRAY OF KrlTokens.Token; IN spaces: ARRAY OF KrlEvents.Space): Context, NEW, ABSTRACT;

(c: Context) **TimeoutMs** (ms: INTEGER): Context, NEW, ABSTRACT;

(c: Context) **TimeoutSec** (ms: INTEGER): Context, NEW, ABSTRACT;

(c: Context) **All** (): Context, NEW, ABSTRACT;

(c: Context) **Cause** (): KrlTokens.Token, NEW, ABSTRACT;

(c: Context) **CauseObj** (): ANYPTR, NEW, ABSTRACT;

(c: Context) **GetCause** (OUT obj: ANYPTR; OUT event: KrlTokens.Token), NEW, ABSTRACT;

(c: Context) **BeginAsync**, NEW, ABSTRACT;

(c: Context) **EndAsync**, NEW, ABSTRACT;

(c: Context) **SetExceptionHandler** (h: KrlLife.Object; handlerContext: ANYPTR), NEW, ABSTRACT;

END;

**ExceptionMsg** = EXTENSIBLE RECORD

context-: Context;

```
code-: INTEGER;
place-: LONGINT;
handlerContext-: ANYPTR
END;
```

END KrcTasks.

DEFINITION **KrlEvents**;

```
IMPORT KrlTokens, KrlLife;
```

```
CONST
  once = TRUE;
```

```
TYPE
```

```
  Space = POINTER TO ABSTRACT RECORD (KrlLife.Object)
    (s: Space) Broadcast (obj: ANYPTR; event: KrlTokens.Token; VAR par:
  ANYREC), NEW, ABSTRACT;
    (s: Space) Cancel (obj: ANYPTR; event: KrlTokens.Token; handler:
  KrlLife.Object; context: ANYPTR), NEW, ABSTRACT;
    (s: Space) Subscribe (obj: ANYPTR; event: KrlTokens.Token; handler:
  KrlLife.Object; context: ANYPTR; once: BOOLEAN; timeoutMs: INTEGER), NEW,
  ABSTRACT
  END;
```

```
  Msg = ABSTRACT RECORD
    obj-: ANYPTR;
    event-: KrlTokens.Token;
    context-: ANYPTR
  END;
```

```
  BroadcastMsg = EXTENSIBLE RECORD (Msg)
    subscribe: BOOLEAN
  END;
```

```
  CancelMsg = EXTENSIBLE RECORD (Msg)
    cause: CancelCauseT
  END;
```

```
VAR
```

```
  gen-: Space;
  stdDir-: Directory;
```

```
PROCEDURE SetDir (d: Directory);
PROCEDURE SetStdSpace (sp: Space);
```

END KrlEvents.

```

MODULE Krl2Http;
  IMPORT
    Tok := KrlTokens;

  TYPE
    SchemeT = POINTER TO EXTENSIBLE RECORD (Tok.EntityT) END;

  VAR
    schemeT-,
    http-, https-: SchemeT;

  TYPE
    MethodT = POINTER TO EXTENSIBLE RECORD (Tok.EntityT) END;

  VAR
    methodT-,
    GET-, POST-, HEAD-, OPTIONS-, PUT-, PATCH-,
    DELETE-, TRACE-, CONNECT-: MethodT;

  TYPE
    HeaderT = POINTER TO EXTENSIBLE RECORD (Tok.EntityT) END;

  VAR
    headerT-,
    Content_Type-, User_Agent-: HeaderT;

  CONST
    www_form_urlencoded* = "application/x-www-form-urlencoded;
charset=utf-8";

END Krl2Http.

```

```

DEFINITION Krl2HttpClient;

```

```

  IMPORT KrlLife, KrlTokens, KrlEvents, Krl2Http, KrcData, KrlText;

  TYPE
    ErrorT = POINTER TO EXTENSIBLE RECORD (KrlTokens.ErrorT) END;
  VAR
    errorT-, connectionErr-, wrongResponseErr-: KrlTokens.ErrorT;

  TYPE
    StageT = POINTER TO EXTENSIBLE RECORD (KrlEvents.StageEv) END;
  VAR
    stageT-, initial-, connecting-, sending-, sent-, receivingHeaders-,
receivingBody-, completed-, failed-: StageT;

    Directory = POINTER TO ABSTRACT RECORD
      (d: Directory) NewRequest (): Request, NEW, ABSTRACT

```

END;

Request = POINTER TO ABSTRACT RECORD (KrlLife.Object)

**troubles**: ARRAY 8 OF KrlTokens.Token;

(r: Request) **Stage** (): StageT, NEW, ABSTRACT;

(r: Request) **Open** (method: Krl2Http.MethodT; scheme:  
Krl2Http.SchemeT;

IN host, uri: ARRAY OF CHAR): Request, NEW, ABSTRACT;

(r: Request) **Header** (header: Krl2Http.HeaderT; IN value: ARRAY OF  
SHORTCHAR): Request, NEW, ABSTRACT;

(r: Request) **Headers** (rd: KrcData.Reader): Request, NEW, ABSTRACT;

(r: Request) **SendReceive** (rd: KrcData.Reader; hwr, cwr:  
KrcData.Writer), NEW, ABSTRACT;

(r: Request) **SendChunk** (rd: KrcData.Reader), NEW, ABSTRACT;

(r: Request) **Receive** (hwr, cwr: KrcData.Writer), NEW, ABSTRACT;

(r: Request) **Receive1xx** (hwr: KrcData.Writer), NEW, ABSTRACT;

(r: Request) **Cancel**, NEW, ABSTRACT;

(r: Request) **Status** (): INTEGER, NEW, ABSTRACT;

(r: Request) **ContentLen** (): LONGINT, NEW, ABSTRACT;

END;

VAR

dir-, stdDir-: Directory;

stdDir-: Directory;

PROCEDURE **DoRequest** (IN reqObj: ARRAY OF CHAR);

PROCEDURE **GetHeader** (rd: KrcData.Reader; header: KrlTokens.EntityT; OUT  
val: ARRAY OF CHAR);

PROCEDURE **GetHeaderL** (rd: KrcData.Reader; header: KrlTokens.EntityT;  
txt: KrlText.Formatter);

PROCEDURE **SetDir** (d: Directory);

END Krl2HttpClient.

PROCEDURE **Task\_GetContent\***;

VAR self: Tasks.Context;

i: INTEGER;

stage: Htc.StageT;

req: Htc.Request;

head, body: Buffers.Kit;

tid: LONGINT;

BEGIN

**Tasks.Begin(tid)**;

self := Tasks.I();

head.New(Buffers.dir, {});

```

body.New(Buffers.dir, {});
req := Htc.dir.NewRequest();
req.Open(Ht.GET, Ht.http, "forum.drakon.su",
"/viewtopic.php?f=153&t=3489")
    .SendReceive(NIL, head.wr, body.wr);
self.Await(req, Htc.completed, NIL).Await(req, Htc.failed,
NIL).TimeoutSec(15).Pass;
IF self.CauseObj() = NIL THEN
    Log.String("Timeout!"); Log.Ln;
    req.Cancel
ELSE
    stage := req.Stage();
    IF stage.Is(Htc.failed) THEN
        Log.String("Failed at stage ");
        Log.String(Tok.LookSuch(req.troubles, Htc.stageT).name);
        Log.String(", error ");
        Log.String(Tok.LookSuch(req.troubles, Htc.errorT).name);
        Log.Ln
    ELSE
        ASSERT(stage = Htc.completed, 100);
        Log.String("Completed!"); Log.Ln
    END
END;
SD.Decode(Buffers.Limited(head.rd));
SD.Decode(Buffers.Limited(body.rd));
SD.Open("HTTP Response");
head.Recycle;
body.Recycle;
Lf.hm.Recycle(req)
END Begin;

```

См. Krl2HttpClient1

```

PROCEDURE AsyncTask*;
    VAR self: Tasks.Context;
        tid: LONGINT;
        var: POINTER TO RECORD

        END;
BEGIN
    Tasks.Begin(tid);
    NEW(var);
    self := Tasks.I();

    (* Подготовка, обмен с глобальным контекстом *)

    self.BeginAsync(Tasks.shareThread);
    (* Вычисления *)
    ....

```

WHILE ...

**self.Pass**

END;

...

self.**EndAsync**;

self.**BeginAsync**(Tasks.ownThread);

(\* Блокирующий вызов к библиотеке \*)

self.**EndAsync**;

(\* Обмен с глобальным контекстом \*)

Lf.hm.Recycle(var)

END AsyncTask;

Два слова о реализации.

Копирование стеков.

Kernel

**SpecialArea\*** = POINTER TO ABSTRACT RECORD

next: SpecialArea;

opts\*: SET

END;

**SpecialAreaMapHeader\*** = RECORD

pnext: POINTER [untagged] TO RECORD END;

opts\*: SET;

prev\*, next\*: ANYPTR; (\* SpecialArea \*)

dataLen\*: INTEGER; (\* MOD 4 = 0 \*)

END;

**SpecialAreaMap\*** = RECORD [untagged]

h\*: SpecialAreaMapHeader;

map\*: ARRAY 64\*1024\*1024 OF SET

(\* LEN(map) = dataLen DIV 4 DIV 32 + 1 (last be empty if MOD = 0)

\*)

END;

**KrLife**

**hm: HeapManager;**

**HeapManager\*** = RECORD

**Recycle\***: PROCEDURE (VAR obj: S.PTR);

**RecycleN\***: PROCEDURE (VAR a, b, c, d, e: S.PTR);

**Status\***: PROCEDURE (obj: ANYPTR): INTEGER;

**DeclareRecycler\***: PROCEDURE (t: Tok.RecTypeT; recycler: PROCEDURE

(p: ANYPTR));

```

GetTypeStat*: PROCEDURE (t: Tok.RecTypeT; OUT defined: BOOLEAN;
VAR stat: ANYREC);
Collect*: PROCEDURE (mode: SET);
CollectTypes*: PROCEDURE (mode: SET; IN types: ARRAY OF
Tok.RecTypeT);
SetStockVolume*: PROCEDURE (t: Tok.RecTypeT; volume: INTEGER);
StockVolumeOf*: PROCEDURE (t: Tok.RecTypeT): INTEGER;
ModesOfType*: PROCEDURE (t: Tok.RecTypeT): SET;
SetTypeModes*: PROCEDURE (t: Tok.RecTypeT; modes: SET);
SetMemModes*: PROCEDURE (x: SET);
MemModes*: PROCEDURE (): SET;
SetNotifyProc*: PROCEDURE (proc: PROCEDURE (event, adr, ret:
INTEGER));
GetNotifyProc*: PROCEDURE (OUT proc: PROCEDURE (event, adr, ret:
INTEGER))
END;

```

```

Wrapper* = POINTER TO ABSTRACT RECORD (View)
  content*: Ref
END;

```

```

wrapper.Set(w.content, myView)

```

Возможности для инкапсулированной в модуле динамической кухни:

### **POINTER [untagged] TO ТипКП**

Динамическая активация untagged (путём правки Kernel.Type.ptroffs).

!! Про отключение сборщика на время цикла-запроса.

### **Лирика**

Подход: защищаемся от ошибки и от раздолбайства (опасные средства локализуются).

Защиту от злонамеренности на обычном железе в едином адресном пространстве не обеспечить.

Вообще, провал идеи "рынка бинарных компонентов" (которую под влиянием MS пропагандировали

в 90-х Oberon Microsystems. Где тот Microsoft COM??

**END Лирика**

### **THAF - Typed Hierarchical App. Format**

Переработка KrcMarkup.

Реализация THAF - И. Кузьмицкий

VAR

```

person1-, person2-, person1_1-, person1_2-: Tf.TagT;

```

```

name-: Tf.StrT;

```

```

birthYear-: Tf.IntT;

```

(\* passport[] - комментируем, что может быть массив паспортов. \*)

**passport-**, **passport2-**: Tf.TagT;  
**number-**, **issuedBy-**: Tf.StrT;  
**values-**: Tf.TagT;

```
PROCEDURE Init;  
BEGIN  
  Tok.InitTokens(Tok.projId, NIL);  
  person.ReqAttr(name).ReqAttr(birthYear).EndAttrs;  
  person.ReqElem(passport).EndElems;  
  
  passport.ReqAttr(number).ReqAttr(issuedBy).EndAttrs;  
  passport.Elem(passport).EndElems;  
  (* Корневой passport - сам массив,  
    он содержит элементы passport *)  
END Init;
```

```
PROCEDURE Do*;  
  VAR doc: Tf.Doc;  
      buf: KrlText.Buffer;  
      par: RECORD END;  
BEGIN  
  doc := Tf.dir.New({}, par);  
  doc.Wr()  
  .O(person1)  
  .Str(name, Tf.post, "Вася Пупкин")  
  .Int(birthYear, Tf.post, 1991)  
  .O(person1_1)  
  .Str(name, Tf.post, "Вася Пупкин1-1")  
  .Int(birthYear, Tf.post, 1981)  
  .X(person1_1)  
  .O(person1_2)  
  .Str(name, Tf.post, "Вася Пупкин1-2")  
  .Int(birthYear, Tf.post, 1982)  
  .X(person1_2)  
  .Ot(passport, Tf.arrT)  
  .O(passport)  
  .Str(name, Tf.post, "5406 999999")  
  .X(passport)  
  .O(passport2)  
  .Str(issuedBy, Tf.post, "Советским РОВД")  
  .X(passport2)  
  .X(passport)  
  .Ot(values, Tf.arrT)  
  .Int(birthYear, Tf.post, 1981)  
  .Int(birthYear, Tf.post, 1981)  
  .Int(birthYear, Tf.post, 1981)  
  .X(values)  
  .X(person1)  
  .O(person2)  
  .Str(name, Tf.post, "Вася Пупкин 2")
```



```

    .Int(birthYear, Tf.post, 1992)
.X(person2)
.Z;
(* вывод в JSON содержимого документа
см.валидатор http://jsonlint.com *)
KrlText.GetTmp(buf, 4000);
KrcThafJson.Export(doc.Rd(), format, buf);
(* Сбрасываем строку-буфер в журнал *)
Log.Ln; Log.String(buf.AsString()); Log.Ln;
(* зачистка *)
KrlText.FreeTmp(buf);
KrlLife.hm.Recycle(doc)
END Do;

```

DEFINITION KrcThaf;

IMPORT KrlTokens, KrlLife, KrcData;

TYPE

```

TagT = POINTER TO EXTENSIBLE RECORD (KrlTokens.EntityT)
  constraints-: SET;
  (t: TagT) Attr (at: KrlTokens.FieldT): TagT, NEW;
  (t: TagT) AttrEx (at: KrlTokens.FieldT; opts: SET): TagT, NEW;
  (t: TagT) Constrain (con: SET), NEW;
  (t: TagT) Elem (et: TagT): TagT, NEW;
  (t: TagT) ElemEx (et: TagT; opts: SET): TagT, NEW;
  (t: TagT) EndAttrs, NEW;
  (t: TagT) EndElems, NEW;
  (t: TagT) GetAttrs (OUT n: INTEGER; OUT attrs: ARRAY OF
KrlTokens.FieldT; OUT opts: ARRAY OF SET), NEW;
  (t: TagT) GetElems (OUT n: INTEGER; OUT elems: ARRAY OF TagT; OUT
opts: ARRAY OF SET), NEW;
  (t: TagT) HasAttr (at: KrlTokens.FieldT): BOOLEAN, NEW;
  (t: TagT) HasElem (et: TagT): BOOLEAN, NEW;
  (t: TagT) LookAttr (IN name: ARRAY OF CHAR): KrlTokens.FieldT, NEW;
  (t: TagT) LookElem (IN name: ARRAY OF CHAR): TagT, NEW;
  (t: TagT) ReqAttr (at: KrlTokens.FieldT): TagT, NEW;
  (t: TagT) ReqElem (et: TagT): TagT, NEW
END;

```

```

Writer = POINTER TO ABSTRACT RECORD (KrlLife.Object)
  (wr: Writer) ArrayFrom (attr: KrlTokens.FieldT; i: INTEGER; replace:
BOOLEAN; rd: KrcData.Reader): Writer, NEW, ABSTRACT;
  (wr: Writer) Attr (attr: KrlTokens.FieldT; i: INTEGER; IN val: ANYREC):
Writer, NEW, ABSTRACT;
  (wr: Writer) Bool (attr: KrlTokens.BoolT; i: INTEGER; x: BOOLEAN):
Writer, NEW, ABSTRACT;
  (wr: Writer) Bools (attr: KrlTokens.BoolT; i: INTEGER; replace:
BOOLEAN; IN x: ARRAY OF BOOLEAN; from, n: INTEGER): Writer, NEW,
ABSTRACT;

```

(wr: Writer) **Byte** (attr: KrlTokens.ByteT; i: INTEGER; x: BYTE): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Bytes** (attr: KrlTokens.ByteT; i: INTEGER; replace: BOOLEAN; IN x: ARRAY OF BYTE; from, n: INTEGER): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Clone** (): Writer, NEW, ABSTRACT;  
 (wr: Writer) **CloneLocal** (): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Data** (IN data: ARRAY OF BYTE; beg, len: INTEGER), NEW, ABSTRACT;  
 (wr: Writer) **Doc** (): Doc, NEW, ABSTRACT;  
 (wr: Writer) **Int** (attr: KrlTokens.IntT; i, x: INTEGER): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Ints** (attr: KrlTokens.IntT; i: INTEGER; replace: BOOLEAN; IN x: ARRAY OF INTEGER; from, n: INTEGER): Writer, NEW, ABSTRACT;  
 (wr: Writer) **K** (key: LONGINT): Writer, NEW, ABSTRACT;  
 (wr: Writer) **KB** (len: INTEGER; IN bytes: ARRAY OF BYTE): Writer, NEW, ABSTRACT;  
 (wr: Writer) **KBR** (IN key: ANYREC): Writer, NEW, ABSTRACT;  
 (wr: Writer) **KS** (IN key: ARRAY OF CHAR): Writer, NEW, ABSTRACT;  
 (wr: Writer) **LInt** (attr: KrlTokens.LIntT; i: INTEGER; x: LONGINT): Writer, NEW, ABSTRACT;  
 (wr: Writer) **LInts** (attr: KrlTokens.LIntT; i: INTEGER; replace: BOOLEAN; IN x: ARRAY OF LONGINT; from, n: INTEGER): Writer, NEW, ABSTRACT;  
 (wr: Writer) **O** (tag: TagT): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Ok** (tag: TagT; key: LONGINT): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Opts** (): SET, NEW, ABSTRACT;  
 (wr: Writer) **Ot** (tag0, tag1: TagT): Writer, NEW, ABSTRACT;  
 (wr: Writer) **OtX** (tag0, tag1: TagT): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Ptr** (attr: KrlTokens.PtrT; i: INTEGER; x: ANYPTR): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Ptrs** (attr: KrlTokens.PtrT; i: INTEGER; replace: BOOLEAN; IN x: ARRAY OF ANYPTR; from, n: INTEGER): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Real** (attr: KrlTokens.RealT; i: INTEGER; x: REAL): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Reals** (attr: KrlTokens.RealT; i: INTEGER; replace: BOOLEAN; IN x: ARRAY OF REAL; from, n: INTEGER): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Ref** (attr: RefT; i: INTEGER; rd: Reader): Writer, NEW, ABSTRACT;  
 (wr: Writer) **RefId** (attr: RefT; i: INTEGER; id: LONGINT): Writer, NEW, ABSTRACT;  
 (wr: Writer) **RefIds** (attr: RefT; i: INTEGER; replace: BOOLEAN; IN x: ARRAY OF LONGINT; from, n: INTEGER): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Refs** (attr: RefT; i: INTEGER; replace: BOOLEAN; IN x: ARRAY OF Reader; from, n: INTEGER): Writer, NEW, ABSTRACT;  
 (wr: Writer) **SInt** (attr: KrlTokens.SIntT; i: INTEGER; x: SHORTINT): Writer, NEW, ABSTRACT;  
 (wr: Writer) **SReal** (attr: KrlTokens.SRealT; i: INTEGER; x: SHORTREAL): Writer, NEW, ABSTRACT;  
 (wr: Writer) **SReals** (attr: KrlTokens.SRealT; i: INTEGER; replace: BOOLEAN; IN x: ARRAY OF SHORTREAL; from, n: INTEGER): Writer, NEW, ABSTRACT;

(wr: Writer) **SStr** (attr: KrlTokens.SStrT; i: INTEGER; IN s: ARRAY OF SHORTCHAR): Writer, NEW, ABSTRACT;  
 (wr: Writer) **SStrPart** (attr: KrlTokens.SStrT; i, pos: INTEGER; replace: BOOLEAN; IN s: ARRAY OF SHORTCHAR): Writer, NEW, ABSTRACT;  
 (wr: Writer) **STxt** (IN txt: ARRAY OF CHAR): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Set** (attr: KrlTokens.SetT; i: INTEGER; x: SET): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Sets** (attr: KrlTokens.SetT; i: INTEGER; replace: BOOLEAN; IN x: ARRAY OF SET; from, n: INTEGER): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Sints** (attr: KrlTokens.SIntT; i: INTEGER; replace: BOOLEAN; IN x: ARRAY OF SHORTINT; from, n: INTEGER): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Str** (attr: KrlTokens.StrT; i: INTEGER; IN s: ARRAY OF CHAR): Writer, NEW, ABSTRACT;  
 (wr: Writer) **StrFrom** (attr: KrlTokens.AnyStrT; i: INTEGER; rd: KrcData.Reader): Writer, NEW, ABSTRACT;  
 (wr: Writer) **StrPart** (attr: KrlTokens.StrT; i, pos: INTEGER; replace: BOOLEAN; IN s: ARRAY OF CHAR): Writer, NEW, ABSTRACT;  
 (wr: Writer) **StrPartFrom** (attr: KrlTokens.AnyStrT; i, pos: INTEGER; replace: BOOLEAN; rd: KrcData.Reader): Writer, NEW, ABSTRACT;  
 (wr: Writer) **T** (tag: TagT): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Tok** (attr: KrlTokens.TokenFieldT; i: INTEGER; x: KrlTokens.Token): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Tokens** (attr: KrlTokens.TokenFieldT; i: INTEGER; replace: BOOLEAN; IN x: ARRAY OF KrlTokens.Token; from, n: INTEGER): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Txt** (IN txt: ARRAY OF CHAR): Writer, NEW, ABSTRACT;  
 (wr: Writer) **X** (tag: TagT): Writer, NEW, ABSTRACT;  
 (wr: Writer) **Xt** (tag0, tag1: TagT): Writer, NEW, ABSTRACT  
 END;

**Doc** = POINTER TO ABSTRACT RECORD (KrlLife.Object)

(d: Doc) **DelAttr** (at: Reader; attr: KrlTokens.FieldT): Doc, NEW, ABSTRACT;  
 (d: Doc) **DelAttrSeq** (at: Reader; attr: KrlTokens.FieldT; beg, end: INTEGER): Doc, NEW, ABSTRACT;  
 (d: Doc) **DelK** (at: Reader): Doc, NEW, ABSTRACT;  
 (d: Doc) **DelKB** (at: Reader): Doc, NEW, ABSTRACT;  
 (d: Doc) **DelKS** (at: Reader): Doc, NEW, ABSTRACT;  
 (d: Doc) **DelSeq** (beg, end: Reader): Doc, NEW, ABSTRACT;  
 (d: Doc) **DelStrSeq** (at: Reader; attr: KrlTokens.FieldT; i, beg, end: INTEGER): Doc, NEW, ABSTRACT;  
 (d: Doc) **DelSub** (from: Reader; beg, end: INTEGER): Doc, NEW, ABSTRACT;  
 (d: Doc) **DelT** (at: Reader; t: TagT): Doc, NEW, ABSTRACT;  
 (d: Doc) **IsHost** (): BOOLEAN, NEW, ABSTRACT;  
 (d: Doc) **NewLocal** (to: Reader): Doc, NEW, ABSTRACT;  
 (d: Doc) **NewRd** (id: LONGINT; opts: SET): Reader, NEW, ABSTRACT;  
 (d: Doc) **NewWr** (to: Reader; opts: SET): Writer, NEW, ABSTRACT;  
 (d: Doc) **Rd** (): Reader, NEW, ABSTRACT;  
 (d: Doc) **Wr** (): Writer, NEW, ABSTRACT

END;

**Reader** = POINTER TO ABSTRACT RECORD (KrlLife.Object)

**res**: SET;

(rd: Reader) **ACnt** (attr: KrlTokens.FieldT): INTEGER, NEW, ABSTRACT;

(rd: Reader) **ADump** (attr: KrlTokens.FieldT; wr: KrcData.Writer): Reader, NEW, ABSTRACT;

(rd: Reader) **AGet** (attr: KrlTokens.FieldT; i: INTEGER; VAR val: ANYREC): Reader, NEW, ABSTRACT;

(rd: Reader) **ABool** (attr: KrlTokens.BoolT; default: BOOLEAN): BOOLEAN, NEW, ABSTRACT;

(rd: Reader) **AByte** (attr: KrlTokens.ByteT; default: BYTE): BYTE, NEW, ABSTRACT;

(rd: Reader) **AGetBools** (attr: KrlTokens.BoolT; from, cnt, to: INTEGER; OUT arr: ARRAY OF SHORTREAL): Reader, NEW, ABSTRACT;

(rd: Reader) **AGetBytes** (attr: KrlTokens.ByteT; from, cnt, to: INTEGER; OUT arr: ARRAY OF BYTE): Reader, NEW, ABSTRACT;

(rd: Reader) **AGetInts** (attr: KrlTokens.IntT; from, cnt, to: INTEGER; OUT arr: ARRAY OF INTEGER): Reader, NEW, ABSTRACT;

(rd: Reader) **AGetLInts** (attr: KrlTokens.LIntT; from, cnt, to: INTEGER; OUT arr: ARRAY OF LONGINT): Reader, NEW, ABSTRACT;

(rd: Reader) **AGetPtrs** (attr: KrlTokens.PtrT; from, cnt, to: INTEGER; OUT arr: ARRAY OF ANYPTR): Reader, NEW, ABSTRACT;

(rd: Reader) **AGetReals** (attr: KrlTokens.RealT; from, cnt, to: INTEGER; OUT arr: ARRAY OF SHORTREAL): Reader, NEW, ABSTRACT;

(rd: Reader) **AGetSInts** (attr: KrlTokens.SIntT; from, cnt, to: INTEGER; OUT arr: ARRAY OF SHORTINT): Reader, NEW, ABSTRACT;

(rd: Reader) **AGetSReals** (attr: KrlTokens.SRealT; from, cnt, to: INTEGER; OUT arr: ARRAY OF SHORTREAL): Reader, NEW, ABSTRACT;

(rd: Reader) **AGetSStr** (attr: KrlTokens.StrT; i: INTEGER; OUT len: INTEGER; OUT s: ARRAY OF SHORTCHAR): Reader, NEW, ABSTRACT;

(rd: Reader) **AGetSets** (attr: KrlTokens.SetT; from, cnt, to: INTEGER; OUT arr: ARRAY OF SET): Reader, NEW, ABSTRACT;

(rd: Reader) **AGetStr** (attr: KrlTokens.StrT; i: INTEGER; OUT len: INTEGER; OUT s: ARRAY OF CHAR): Reader, NEW, ABSTRACT;

(rd: Reader) **AGetTokens** (attr: KrlTokens.TokenFieldT; from, cnt, to: INTEGER; OUT arr: ARRAY OF KrlTokens.Token): Reader, NEW, ABSTRACT;

(rd: Reader) **AIInt** (attr: KrlTokens.IntT; default: INTEGER): INTEGER, NEW, ABSTRACT;

(rd: Reader) **ALIInt** (attr: KrlTokens.LIntT; default: LONGINT): LONGINT, NEW, ABSTRACT;

(rd: Reader) **AOpen** (attr: KrlTokens.FieldT): KrcData.Reader, NEW, ABSTRACT;

(rd: Reader) **APtr** (attr: KrlTokens.PtrT; default: ANYPTR): ANYPTR, NEW, ABSTRACT;

(rd: Reader) **AReal** (attr: KrlTokens.RealT; default: REAL): REAL, NEW, ABSTRACT;

(rd: Reader) **ASInt** (attr: KrlTokens.SIntT; default: SHORTINT): SHORTINT, NEW, ABSTRACT;

(rd: Reader) **ASReal** (attr: KrlTokens.SRealT; default: SHORTREAL):

```

SHORTREAL, NEW, ABSTRACT;
    (rd: Reader) ASet (attr: KrlTokens.SetT; default: SET): SET, NEW,
ABSTRACT;
    (rd: Reader) AStrLen (attr: KrlTokens.StrT; i: INTEGER): INTEGER, NEW,
ABSTRACT;
    (rd: Reader) AToken (attr: KrlTokens.TokenFieldT; default:
KrlTokens.Token): KrlTokens.Token, NEW, ABSTRACT;
    (rd: Reader) Beg (): Reader, NEW, ABSTRACT;
    (rd: Reader) ByRef (attr: RefT; i: INTEGER): Reader, NEW, ABSTRACT;
    (rd: Reader) Clone (): Reader, NEW, ABSTRACT;
    (rd: Reader) CloneLocal (): Reader, NEW, ABSTRACT;
    (rd: Reader) EBndLev (): INTEGER, NEW, ABSTRACT;
    (rd: Reader) EGetAttrs (OUT n: INTEGER; OUT attrs: ARRAY OF
KrlTokens.Token): Reader, NEW, ABSTRACT;
    (rd: Reader) EGetKey (OUT key: LONGINT): Reader, NEW, ABSTRACT;
    (rd: Reader) EGetKeyB (OUT len: INTEGER; OUT key: ARRAY OF BYTE):
Reader, NEW, ABSTRACT;
    (rd: Reader) EGetKeyBR (OUT key: ANYREC): Reader, NEW, ABSTRACT;
    (rd: Reader) EGetKeyS (OUT key: ARRAY OF CHAR): Reader, NEW,
ABSTRACT;
    (rd: Reader) EGetTag (i: INTEGER; OUT t: TagT): Reader, NEW,
ABSTRACT;
    (rd: Reader) EGetTags (OUT n: INTEGER; OUT tags: ARRAY OF
KrlTokens.Token): Reader, NEW, ABSTRACT;
    (rd: Reader) EId (): LONGINT, NEW, ABSTRACT;
    (rd: Reader) EKey (): LONGINT, NEW, ABSTRACT;
    (rd: Reader) ELev (): INTEGER, NEW, ABSTRACT;
    (rd: Reader) ETag (i: INTEGER): TagT, NEW, ABSTRACT;
    (rd: Reader) End, NEW, ABSTRACT;
    (rd: Reader) Opts (): SET, NEW, ABSTRACT;
    (rd: Reader) SibCnt (): INTEGER, NEW, ABSTRACT;
    (rd: Reader) SibPos (): INTEGER, NEW, ABSTRACT;
    (rd: Reader) SubCnt (): INTEGER, NEW, ABSTRACT;
    (rd: Reader) ToId (id: LONGINT): Reader, NEW, ABSTRACT;
    (rd: Reader) ToK (key: LONGINT): Reader, NEW, ABSTRACT;
    (rd: Reader) ToKB (len: INTEGER; IN bytes: ARRAY OF BYTE): Reader,
NEW, ABSTRACT;
    (rd: Reader) ToKBR (IN key: ANYREC): Reader, NEW, ABSTRACT;
    (rd: Reader) ToKS (IN key: ARRAY OF CHAR): Reader, NEW, ABSTRACT;
    (rd: Reader) ToSib (n: INTEGER): Reader, NEW, ABSTRACT;
    (rd: Reader) ToSub (): Reader, NEW, ABSTRACT;
    (rd: Reader) ToSup (lev: INTEGER): Reader, NEW, ABSTRACT;
    (rd: Reader) Valid (): BOOLEAN, NEW, ABSTRACT
END;

END KrcThaf.

```

## Архитектура онлайн-сервисов (SelfBoss.ru)

XQuery-подобная надстройка над реляционной.

VAR

```
topic-: Mk.TypeT;  
  (* Queries.id *)  
topicTypeVid-: Mk.LongintT;  
title-, info-: Mk.StringT;  
priceTypeVid-, currencyVid-: Mk.LongintT;  
price-: Mk.RealT;  
pubTime-, editTime-: Mk.LongintT;  
viewsCount-: Mk.IntegerT;  
stateVid-: Mk.LongintT; (* *)  
state2Vid-: Mk.LongintT;  
publicModeVid-, banModeVid-: Mk.LongintT; (* publicMode - сам  
человек переключает. убрал из поиска. ban - закрыла администрация. *)  
editModeVid-: Mk.LongintT;
```

```
Mk.InitFullName(topic);  
topic.Attr(Q.id).Attr(topicTypeVid)  
  .Attr(title).Attr(info).Attr(priceTypeVid).Attr(currencyVid)  
  .Attr(price)  
  .Attr(pubTime).Attr(editTime).Attr(viewsCount)  
  .Attr(stateVid).Attr(state2Vid).Attr(publicModeVid).Attr(banModeVid)  
  .Attr(editModeVid).EndAttrs;  
topic.Elem(project).Elem(vacancy).Elem(service).EndElems;
```

### Запрос к БД:

Фаза SELECT:

```
stat := FncDb.qdir.NewStatements(decl)  
  .Select(Q.L1).From(M.topic);  
IF topicTypeVid = M.projects THEN  
  stat.From(M.project).Z  
ELSIF topicTypeVid = M.vacancies THEN  
  stat.From(M.vacancy).Z  
ELSIF topicTypeVid = M.services THEN  
  stat.From(M.service).Z  
END;  
stat.Eq(M.topicTypeVid, "topicTypeVid").Z;  
  IF ~allGroup OR allGroup & (subjGroupVid # 0) THEN  
    stat.LinkedTo(M.topicSubject, "topicSubjects").Z  
  END;  
  stat.Eq(M.stateVid, "topicState")  
  .Eq(M.publicModeVid, "null")  
  .Eq(M.banModeVid, "null").Z;  
FOR i := 0 TO traits.gn-1 DO  
  stat.LinkedTo(M.topicTrait, "topicTraits" + KrlStrings.num[i]).Z  
END;
```

Результат - один или несколько списков ID.

Фаза FETCH - выборка записей по ID из списков и группировка в иерархическую структуру:

```
selectTopics.fetch := FncDb.qdir.NewFetch(selectTopics.decl);
selectTopics.fetch.Struct()
  .Elc(M.topic, Q.fromList).Str(Q.list, Q.L1)
    .El1(M.project)
    .El1(M.vacancy)
    .El(M.service)
      .Elc(M.serviceOption, Q.outRel)
        .El1(M.option)
        .End(M.serviceOption)
      .End(M.service)
    .Elc1(M.topicSubject, Q.outRel)
    .Elc(MFiles.fileAttachTo, Q.inRel)
      .El1(MFiles.file)
      .End(MFiles.fileAttachTo)
    .Elc1(M.topicTrait, Q.outRel)
    .Elc(M.topicAuthor, Q.outRel)
      .El1(MSoc.person)
      .End(M.topicAuthor)
    .Elc(M.topicPlace, Q.outRel)
      .El(MGeo.place)
        .Elc(MGeo.placeCity, Q.outRel)
          .El1(MGeo.city)
          .End(MGeo.placeCity)
        .End(MGeo.place)
      .End(M.topicPlace)
    .End(M.topic).Z
```

Всё за 1 запрос-ответ к СУБД (использование временных таблиц, возврат мультитабличного результата).

Между сервером UI и ядром - обмен структурированными сообщениями через брокер запросов - единая точка (журнал, безопасность).

-----  
**ФРОНТЭНД**  
-----

**BscPageV - интерфейсный модуль:**

```
Page* = POINTER TO ABSTRACT RECORD (Krl2Views.View)
  title*: TitleStr;
  toolbar*, tabLines*, content*: Views.Ref
END;
```

## BscPageV1 - модуль реализации:

Page = POITE

```
PROCEDURE (p: Page) Gen2 (out: KrcThaf.Writer);
  VAR context: BscmPV.Context;
      cssLink, jsLink, uriFiles: ARRAY 256 OF CHAR;
      uriCl, uriMn: Txt.Buffer;
BEGIN
  KrwStatic.UriToLinked(cssContent, cssLink);
  KrwStatic.UriToLinked(jsContent, jsLink);
  KrwStatic.UriToStatic("Bscm", uriFiles);

  context := p.Scope().mainObj(BscPV.Context);
  out.O(H.html).Str(H.lang, 0, "ru")
    .O(H.head)
      .O(H.meta).Str(H.charset, 0, "utf-8").X(H.meta)
      .O(H.title).Txt(p.title).X(H.title)
      .O(H.link).Str(H.rel, 0, "stylesheet").Str(H.link_type, 0,
"text/css").Str(H.href, 0, cssLink).X(H.link)
      .O(H.link).Str(H.rel, 0, "stylesheet").Str(H.link_type, 0,
"text/css").Str(H.href, 0, uriFiles + "css/simplelightbox.css").X(H.link)
      .O(H.script).Str(H.src, 0, uriFiles + "js/jquery.min.js").X(H.script)
      .O(H.script).Str(H.src, 0, uriFiles + "js/Hyphenator.js").X(H.script)

      .O(H.script).Str(H.src, 0, uriFiles + "js/jquery.jplayer.js").X(H.script)
      .O(H.script).Str(H.src, 0, uriFiles + "js/aurora.js").X(H.script)
      .O(H.script).Str(H.src, 0, uriFiles + "js/mp3.js").X(H.script)
      .O(H.script).Str(H.src, 0, uriFiles + "js/simple-
lightbox.min.js").X(H.script)
      .O(H.script).Str(H.src, 0, uriFiles + "js/scripts.js").X(H.script)
      .O(H.script).Str(H.src, 0, jsLink).X(H.script)
      .O(H.script).Txt("Hyphenator.run();").X(H.script)
    .X(H.head)
    .O(H.body).T(csFilesBody)
      .O(H.div).T(csLogo)
        .O(H.div).T(csImg).X(H.div)
        .O(H.div).T(csLogOut)
          .O(H.div).T(csText).Txt("Выход").X(H.div)
          .O(H.div).T(csLogIco).X(H.div)
        .X(H.div)
      .O(H.div).T(csLk)
        .O(H.div).T(csName).Txt(context.userName).X(H.div)
        .O(H.div).T(csIco).X(H.div)
        .O(H.ul).T(csLkMenu).Z;
          uriCl := Txt.bufDir.New(128);
          Uri.ClientsTab(uriCl);
          uriMn := Txt.bufDir.New(128);
          Uri.ManagersTab(uriMn);
          out.O(H.li).O(H.a).Str(H.href, 0,
```



```

uriCl.AsString()).Txt("Клиенты").X(H.a).X(H.li)
    .O(H.li).O(H.a).Str(H.href, 0,
uriMn.AsString()).Txt("Менеджеры").X(H.a).X(H.li)
    .X(H.ul)
    .X(H.div)
    .X(H.div).Z;

```

```

Lf.hm.Recycle(uriCl);
Lf.hm.Recycle(uriMn);

```

```

context.tabLine.To().Gen(out);
p.toolbar.To().Gen(out);

```

```

IF p.tabLines.to # NIL THEN
    p.tabLines.To().Gen(out);
END;

```

```

out.O(H.div).T(csContent).Z;
    p.content.To().Gen(out);
out.X(H.div)
.X(H.body)
.X(H.html).Z
END Produce;

```

Кэширование, быстроедействие рендеринга (20 мс).  
 Реализация - Павел Снитко.

```

Button* = POINTER TO ABSTRACT RECORD (Views.View)
    label*: Types.ViewTitle;
    uri*: Types.Uri;
END;

```

```

Button = POINTER TO RECORD (V.Button)
    classes: ARRAY 4 OF Tok.Token;
END;

```

```

PROCEDURE (b: Button) Prepare;
BEGIN
    Tok.Incl(b.classes, csButton);
    IF Tok.HasSuch(b.traits, V.active) THEN
        Tok.Incl(b.classes, csAbutton);
    ELSIF Tok.HasSuch(b.traits, V.passive) THEN
        Tok.Incl(b.classes, csPbutton);
    END;
END Prepare;

```

```

PROCEDURE (b: Button) Produce (out: Views.Writer);
    VAR p: Views.GeneratorProc;
BEGIN
    out

```

```
.OVar(H.div).VarTs(b.classes)
  .VarTxt(b.label)
.X(H.div).Z;
END Produce;
```

Отзывы про "безумие фронтэнда" у JS-программистов.  
Желание и блочности, и накладные расходы. 10-12 с.

Изоморфный JavaScript. У нас - тот же подход, но с КП на сервере.  
Сегментное обновление страницы, VirtualDOM.  
Стейт рендеринга, хранимый как Stores.Store.

JS, CSS - по MD5 от содержимого.

Факторы и успеха, и движения дальше

1) И21: Статическая типизация + динамика  
Динамика + эффективность

2) Дистиллированное семант. ядро (и абстракция от машины, и аппарат формализации предметки).

3) IDEE - метаинформация, сама модель метаинформации, на которую можно опереться.

3) Понимая, что лучшая основа DSL - это просто дистиллированное ООП (любое понятие должно жить в коде как тип, любая информация предметки, в идеале, фиксируется статически), эффективно применять инструмент для повышения уровня кода.

Код на КП становится совсем не похож на "просто императивный".

4) Надстраивать любые эксперименты с 4GL\CASE\.. и т.п. ТОЛЬКО НА ПРОЯВЛЕННЫЙ, ВЫЧИЩЕННЫЙ УРОВЕНЬ 3GL-семантики.

Заметим новую угрозу: надстройку нечётких систем принятия решений надо всем этим уровнем (Internet of Things, боевые БПЛА и т.п.)

Недопустимо и опираться на раздутые языки, перегружая их дальше, и просто скрывать императивную семантику "под капот" предметно-ориентированного инструмента (откуда она всё равно потом прорывается в виде жутких скриптовых вылезаний).

Тезис И21: "Программирование - это наиболее общая форма взаимодействия человека с компьютером".

Идея любого CASE на базе ББ: исходник модуля - как комплект бинарных файлов.

Таблица деклараций, ДРАКОН-схема или конечный автомат.

При компиляции исходник собирается "на лету" и подаётся на вход компилятору.

Для рантайма - сгенерённый модуль, плюс вся метаинформация нужного

типа.

5) Всё-таки, получив опыт, можно и делать шаг в реализации нового стека (уже не просто на базе Оберона, что-то ближе к опыту Эль-76 и т.п.), имея рабочий, безотказный стек для обучения, текущих задач - да и для новых инструментов - на базе Оберона.