Настройки компонентов Блэкбокс — соглашения о размещении

Темиргалеев Е. Э.
(ТЕЭ 04.11.2016 22:12:20 +0300, испр. ТЕЭ 04.01.2017 7:51:41 +0300)

1. Соглашения о размещении настроек отсутствуют

Настройка (конфигурация) компонента — его состояние при загрузке. Настройки по-умолчанию (заводские) могут определяться компонентом, могут требовать явной настройки перед эксплуатацией. Перманентные изменения настроек хранятся (должны храниться) вне компонента (*).

Пример, иллюстрирующий оба момента — меню стандартного Блэкбокс (Windows-реализация):
- настройки динамические — *HostMenus* содержит набор команд для формирования меню;
- *StdMenuTool* — формирует их по текстам документов "*Xxx*/Rsrc/Menus", которые представляют заводские настройки меню (как системных, так и прочих компонентов);
- пользовательские настройки вносятся изменением этих документов, нарушая принцип (*).

На этом примере выявляется концептуальная неполнота соглашений Блэкбокс о размещении частей компонента — вопрос о хранении настроек отдельно никак не оговорён. Ещё одной иллюстрацией нарушения (*) является удобное стихийное решение по хранению настроек в текстовых ресурсах [1].

Частично проблема снята помещением настроек в недра централизованного хранилища настроек Windows — реестр. Рудимены — файлы "Dev/Rsrc/AnaOpt.opt" и "Dev/Rsrc/BrowOpt.opt". Понятно, что это решение порождает другую проблему, которую товарищи (сборка «Информатики-21», «Центра Каркаса Блэкбокс») решают отказом от реестра в пользу .ini-файла.

2. Соглашения о настройках

Предлагаемое концептуальное решение — хранить настройки всех компонентов в подсистеме *Config*. Это, во-первых, позволит выделить файлы настроек вовне каталогов компонентов. Во-вторых, даст возможность работы с файлами настроек через штатные инструменты работы с модулями по соглашениям о файлах подсистем [2].

Соглашения:
- для компонента *XxxYyy* (модуля *Yyy* подсистемы *Xxx*) компонент *ConfigGXxxYyy* считается «настроечным»:
  ИмяМодуля = "Config" "G" Xxx Yyy.
- заглавная литера (выбрана "G") требуется для подсистем, имена которых начинаются со строчной литеры (*i21sys*, *ipui*);
- неизменяемые (заводские) настройки (если есть) хранятся в ресурсах *XxxYyy*; настройки пользователя — в одноимённых ресурсах настроечного компонента *ConfigGXxxYyy*:
  "Xxx/Rsrc/YyyTemplate.odc" — заводской шаблон "Template" (язык по-умолчанию)
  "Xxx/Rsrc/en/YyyTemplate.odc" — заводской шаблон "Template" (язык: английский)

"Config/Rsrc/GXxxYyyTemplate.odc" — пользовательский шаблон "Template" на основе заводского с подходящим языком

"Config/Rsrc/GXxxYyyOpt.ini" — пользовательские настройки "Opt.ini"

- заводские настройки системных компонентов лежат в "System":

"System/Rsrc/ZzzCfg.cfg" — заводские

"Config/Rsrc/GZzzCfg.cfg" — пользовательские

Примерная схема чтения файла настроек (записана с учётом дельты B29):

```
ipuiK379.Old("Xxx", "Yyy", "Template", "odc", loc, name, f);
```

⇩

PROCEDURE **Old** (IN sub, mod, key: ARRAY OF CHAR; IN type: Files.Type; OUT loc: Files.Locator; OUT name: Files.Name; OUT f: Files.File)
Открывает текущий файл настроек. Для чтения или проверки наличия.

Постусловие
f # NIL открыт файл настроек
(loc, name) — местоположение текущего файла настроек
loc.res = 0 пользовательский
loc.res = 2 заводской
f = NIL файл настроек не открыт
loc.res # 0 — стандартная ошибка *l*(*Files.Locator*).*res*

PROCEDURE **Old*** (IN sub, mod, key: ARRAY OF CHAR; IN type: Files.Type; OUT loc: Files.Locator; OUT name: Files.Name; OUT f: Files.File);

```
    VAR  locL: Files.Locator;
BEGIN
    Files.dir.GetFileName("G" + sub + mod + key, type, name);
    loc := Files.dir.This(configDir).This(rsrcDir); f := Files.dir.Old(loc, name, Files.shared);
    IF f # NIL THEN
        ASSERT(loc.res = 0, 100)
    ELSE
        Files.dir.GetFileName(mod + key, type, name);
        IF sub = "" THEN loc := Files.dir.This(sysDir) ELSE loc := Files.dir.This(sub) END;
        loc := loc.This(rsrcDir);
        IF Dialog.language # "" THEN
            locL := loc.This(Dialog.language); f := Files.dir.Old(locL, name, Files.shared);
            IF f # NIL THEN loc := locL END
        END;
        IF f = NIL THEN f := Files.dir.Old(loc, name, Files.shared) END;
        ASSERT((f # NIL) = (loc.res = 0), 100);
        IF f # NIL THEN loc.res := 2 END
    END
END Old;
```

⇦

3. Внедрение схемы

3.1 Блэкбокс: *Init* и *Config*, *StdMenuTool* (дельта E08)

Init вызывает пользовательскую команду динамической настройки среды *Config.Setup*. Модуль *Config* считаем настречным для *Init*:

- модуль *Config* можно считать заводскими настройками системы (сборки);
- сначала отрабатывается вызов пользовательских настроек *ConfigGInit.Setup*.

```
PROCEDURE Init;
    VAR res: INTEGER; ...
BEGIN
```

```
    …
    Dialog.Call("ConfigGInit.Setup", "", res);
    IF res # 0 THEN Dialog.Call("Config.Setup", "", res) END;
    HostMenus.Run
  END Init;
```

Модернизированный вариант *StdMenuTool* выполнен в *ipuiK378*:

- заводские настройки меню читаются по исходной схеме из "*Xxx*/Rsrc/Menus.odc"
- пользовательские настройки меню считаются настройками компонента *ipuiK378* (или *StdMenuTool*):

"Config/Rsrc/GipuiK378System.odc"
"Config/Rsrc/GipuiK378Dev.odc"
"Config/Rsrc/GipuiK378Text.odc"
…

В *Init* в порядке обратной совместимости вызывается и *StdMenuTool.UpdateAllMenus*:

```
  PROCEDURE Init;
    VAR res: INTEGER; …
  BEGIN
    …
    Dialog.Call("ipuiK378.UpdateAllMenus", "", res);
    IF res # 0 THEN Dialog.Call("StdMenuTool.UpdateAllMenus", "", res) END;
    …
  END Init;
```

3.2 Настроечные модули

Схема [4] применяется в *ipuiK22*/*ipuiK23*:

```
  Meta.LookupPath("~~ipuiK23~~ConfigGipuiK22.Setup", item); …
```

Правки *ipuiK22*:

- имя заменено;
- шаблон настроечного модуля — в документацию.

Регулярно используемые псевдонимы должна задавать команда *Setup* модуля настройки *ConfigGipuiK22*, активируемая при инициализации. Тогда довольно вызова *ipuiK22.Install*.

```
!"ipuiK298.WriteThis('Config/Mod', 'GipuiK22', '')"
MODULE ConfigGipuiK22;

  IMPORT K := ipuiK22;

  PROCEDURE Setup*;
  BEGIN
    K.SetMapping('StdMenuTool', 'ipuiK378');
    K.SetMapping('ert0devCommanders', 'ipuiK297');
    K.SetMapping('ert0devDebug', 'ipuiK300');
  END Setup;

END ConfigGipuiK22.
!DevCompiler.CompileThis ConfigGipuiK22
!ipuiK22.Install !StdMenuTool.ListAllMenus !ipuiK22.Uninstall
```

Схема [4] применяется в *ipuiK173* (настроечный *ipuiK188*). Переделка к соглашениям: вызывать пользовательский настроечный, если «хук» не установлен, вызывать установку заводского «хука»:

```
PROCEDURE Init;
   VAR    res: INTEGER;
BEGIN
   Dialog.Call("ConfigGipuiK173.Setup", "", res);
   IF (res # 0) OR (hook = NIL) THEN
      Dialog.Call("ipuiK174.Install", " ", res)
   END
END Init;
```

Настроечный *ipuiK188* упразднён, в *ipuiK174* добавлена команда *Install* (обычная практика).

3.3 Текстовые ресурсы как хранилище настроек

Данное (концептуально неверное) применение для текстовых ресурсов становится допустимым:
- настройки пользователя хранятся в текстовых ресурсах настроечного компонента;
- заводские настройки — в текстовых ресурсах самого [3].

Модификация *ipuiK20*: выставлен точный результат запроса цепочки, глобальная
VAR **result***: INTEGER
Выставляется в *MapString* и *MapParamString* как результат *ipuiK20.MapString1*. Позволяет точно проверить факт наличия запрошенной цепочки в таблице.

Схема чтения значения из настроек:
ipuiK380.GetValue("Xxx", "Yyy", "Value", value, res)

⇩

PROCEDURE **GetValue** (IN sub, mod, key: ARRAY OF CHAR; OUT value: ARRAY OF CHAR; OUT res: INTEGER)
Читает значение *key* для компонента *SubMod* в цепочку *value*.

Предусловие
20 *submod* — идентификатор
21 заводское значение *key* влезло в *value*

Постусловие
res = 0 пользовательское значение
res = 1 пользовательское значение, целиком не влезло
res = 2 заводское значение
res = 3 не определено

```
IMPORT LS := ipuK20;

PROCEDURE GetValue* (IN sub, mod, key: ARRAY OF CHAR; OUT value: ARRAY OF CHAR; OUT res: INTEGER);
BEGIN
   LS.MapString("#ConfigG" + sub + mod + ":" + key, value);
   ASSERT(LS.result # 3, 20);    (* submod — идентификатор *)
   ASSERT(LS.result IN {0, 4, 5, 6}, 100);
   IF LS.result = 0 THEN
      res := 0    (* настройка пользователя *)
   ELSIF LS.result = 6 THEN
      res := 1    (* настройка пользователя, значение не влезло в value *)
   ELSE
      LS.MapString("#" + sub + mod + ":" + key, value);
      ASSERT(LS.result # 6, 21);    (* заводское значение влезло в value *)
      ASSERT(LS.result IN {0, 4, 5}, 101);
      IF LS.result = 0 THEN
         res := 2    (* заводская настройка *)
      ELSE
         value := ""; res := 3    (* значение не определено *)
      END
```

```
    END
  END GetValue;
```

Компоненты *ipuiK171* и *ipuiK293* используют подобную схему, читая настройки из текстовых ресурсов «настроечных» компонентов *ipuiK183* и *ipuiK295*.
Переделка:
- содержимое текстовых ресурсов «вернуть» в ресурсы компонента как заводские настройки;
- сменить алгоритм чтения.

ipuiK293

Mod

```
  PROCEDURE NewDocFile (IN subName, subSuff, cat, modName, modSuff: ARRAY OF CHAR; OUT result: INTEGER);
  …
      IF code = 0 THEN
      LS.MapString("#ipuiK295:" + procKey, cmd);
      ipuiK380.GetValue("ipui", "K293", procKey, cmd, res);
      IF res IN {1, 3}cmd = procKey THEN
        …
      ELSE
        cmd := cmd + "('" + subName + "', '" + subSuff + "', '" + cat + "', '" + modName + "', '" + modSuff + "')";
        DialogCall(cmd, modTextResKey + "НеВыпКомандаСозданияФайла", res);
        IF res # 0 THEN code := 4 END;
        …
      END
    END;
```

Docu

2.1 Генератор документов

Для создания документов вызывается процедура

```
  PROCEDURE ГенДок (IN subName, subSuff, cat, modName, modSuff: ARRAY OF CHAR)
```

Её можно заменить, задав соотв. значение настройке "ПроцГенДок":

❗"ipuiK298.WriteThis('Config/Rsrc', 'GipuiK293Strings', '')"

ПроцГенДок ipuiK294.Do

RStrings

…

ПроцГенДок ipuiK294.Do

ipuiK171

Mod

```
  CONST
    …
    configMod = "#ipuiK183:";
  …
  PROCEDURE GetColor (IN cmd: ARRAY OF CHAR; OUT defined: BOOLEAN; OUT color: INTEGER);
    VAR   s: TextMappers.Scanner; qi: ARRAY 256 OF CHAR; ok: BOOLEAN;
      colStr: ARRAY 300 OF CHAR; x, res: INTEGER;
  BEGIN
```

```
    (* кощунственно,  но... моску  ужо не до красоты,  а сборщик мусора  переживёт *)
    s.ConnectTo(TextModels.dir.NewFromString(cmd)); s.Scan;
    ok := s.type = TextMappers.string;
    IF ok THEN TextMappers.ScanQualIdent(s, qi, ok) END;
    IF ok THEN
      LS.MapString(configMod + qi, colStr);
      ipuiK380.GetValue("ipui", "K171", qi, colStr, res); ok := res IN {0, 2};
      IF ok THEN Strings.StringToInt(colStr, x, res); ok := res = 0 END
    END;
    defined := ok;
    IF ok THEN color := x ELSE color := Ports.defaultColor END
  END GetColor;
```

Docu

- цвет вьюшек-заплаток настраивается в текстовых ресурсах настроечного модуля *ConfigGipuiK171*. Ключ — команда, значение — числовой код цвета. Например:

```
"ipuiK298.WriteThis('Config/Rsrc', 'GipuiK171Strings', '')"
ipuiK172.Insert    000FF00H
ipuiK172.Delete    00000FFH
ipuiK172.Replace    0FF0000H
```

RStrings

```
…
ipuiK172.Insert    000FF00H
ipuiK172.Delete    00000FFH
ipuiK172.Replace    0FF0000H
```

Аналогично применено к *ipuiK202* (*omcCmdline*).

3.4 Настраиваемые шаблонные документы

Ресурсный документ хранит используемый компонентом шаблон. Настройка предполагает изменение ресурса — нарушает (*). Применение соглашений о настройках:

- заводская версия документа-шаблона хранится в ресурсах компонента;
- изменённый (настроенный) пользователем шаблон — в ресурсах настроечного компонента.

Примеры:
ipuiK294

```
  PROCEDURE GetTemplate (IN cat: ARRAY OF CHAR; OUT template: TextModels.Model);
    VAR    loc: Files.Locator; name: Files.Name; v: Views.View; f: Files.File;
  BEGIN
    name := "K294" + cat;
    loc := Files.dir.This("ipui").This("Rsrc");
    IF loc.res = 0 THEN v := Views.OldView(loc, name) END;
    ipuiK379.Old("ipui", "K294", cat, Files.docType, loc, name, f);
    IF f # NIL THEN v := Views.OldView(loc, name) END;
    IF loc.res # 0 THEN
      …
      ShowThisLocErrMsg(loc.res, "#ipuiK294:ДокНеПрочитан", name, "")
    ELSIF ~(v IS TextViews.View) THEN
      template := NIL;
```

```
      DialogShowParamMsg("#ipuiK294:ДокНеТекстовый", name, "", "")
    ELSE
      template := Models.CopyOf(v(TextViews.View).ThisModel())(TextModels.Model)
    END
  END GetTemplate;
```

ipuiK293

```
  PROCEDURE ReadTemplate (IN key: ARRAY OF CHAR; OUT code, res: INTEGER; OUT v: Views.View);
    VAR   loc: Files.Locator; name: Files.Name; f: Files.File;
  BEGIN
    ipuiK379.Old("ipui", "K293", key, Files.docType, loc, name, f);
    IF f # NIL THEN v := Views.OldView(loc, name) END;
    v := Views.OldView(loc, name$);
    IF v = NIL THEN   (* постусловия ReadText *)
      code := 1; res := loc.res
    ELSIF ~(v IS TextViews.View) THEN
      code := 2
    ELSE
      code := 0
    END
  END ReadTemplate;

  PROCEDURE GetTemplate (OUT template: TextModels.Model);
    VAR
      sub, name: ARRAY 128 OF CHAR;
      code, res: INTEGER; v: Views.View;
      fmt: TextMappers.Formatter;
  BEGIN
    IF extMode THEN
      sub := subDevName$; name := "Dev-Map-K293";
      ReadText(sub, name, rsrc, code, res, v);
      IF (code = 1) & (res = 2) THEN
        sub := "ipui"; name := "K293dev";
        ReadText(sub, name, rsrc, code, res, v)
        ReadTemplate("dev", code, res, v)
      END
    ELSE
      sub := "ipui" ; name := "K293";
      ReadText(sub, name, rsrc, code, res, v)
      ReadTemplate("", code, res, v)
    END;
    IF code = 0 THEN
      template := Models.CopyOf(v(TextViews.View).ThisModel())(TextModels.Model)
    ELSE
      template := TextModels.dir.New();
      …
    END;
    ASSERT(template # NIL, 60)
  END GetTemplate;
```

Список источников

1. Темиргалеев Е. Э. Заметки о формировании сборок и компонентах Блэкбокс // URL: http://oberoncore.ru/library/temir_zametki_o_formirovanii_sborok_i_komponentax_ble_kboks
2. Темиргалеев Е. Э. Разработочная подсистема — упорядочивание сопутствующих процессу разработки материалов // URL: http://oberoncore.ru/library/temir_razrabotochnaya_podsistema-uporyadochivanie_soputstvuyushhix_processu_razrabotki_materialov
3. Темиргалеев Е. Э. Определение текстовых ресурсов на уровне компонента Блэкбокс // URL: http://oberoncore.ru/library/temir_opredelenie_tekstovy_x_resursov_na_urovne_komponenta_ble_kboks
4. Темиргалеев Е. Э. Настроечный модуль для обеспечения готовности компонента по загрузке // URL: http://oberoncore.ru/library/temir_nastroechny_j_modul_dlya_obe specheniya_gotovnosti_komponenta_po_zagruzke