Developer Camp

【B2】C++Builderテクニカルセッション

「いまどきのC++開発をもっと楽にする3つの武器」

NTTデータ ビジネスブレインズ
シニア・スペシャリスト　伊賀敏樹

はじめに

はじめに

発表内容については私自身の見解であり、必ずしも所属企業および所属組織における立場、戦略、意見を代表するものではありません。

アジェンダ

- はじめに
- 自己紹介
- モバイルデバイスの普及
- C++Builder の簡易な紹介
- コンポーネント(1つめ)
- 言語の機能(2つめ)
- マルチデバイスのサポート(3つめ)
- まとめ
- Q&A

自己紹介

自己紹介【業務】

- NTTデータビジネスブレインズに勤務
  - 多種多様なシステム構築に従事。
  - 方式屋、フレームワーカー
  - 実プロジェクトにプログラマーとしての投入

- NTTデータビジネスブレインズについて

  NTTデータビジネスブレインズは、2003年にNTTデータおよび日本板硝子による 株式構成によって設立された、ITを活用したトータルソリューションを提供する会社です。さまざまな情報システム構築にたずさわり、またパッケージソフト の開発および販売をおこなうなど、多様なビジネスを展開しています。日本板硝子におけるシステム構築で培ったノウハウとNTTデータの高い技術力を武器に、法人顧客を中心としてビジネスを拡大していっております。 詳細は、www.nttd-bb.com をご覧ください。

自己紹介【ライター】

- ブロガー
  いがぴょんの日記
  http://www.igapyon.jp/igapyon/
  http://d.hatena.ne.jp/igapyon/

- テクニカル・ライター
  書籍 / 雑誌 / Web記事の執筆

いま

- いがぴょんの『C++言語入門 ～ C++Builderでビジュアルプログラミング』
  http://www.embarcadero.com/jp/cbfan/cpp-lang/

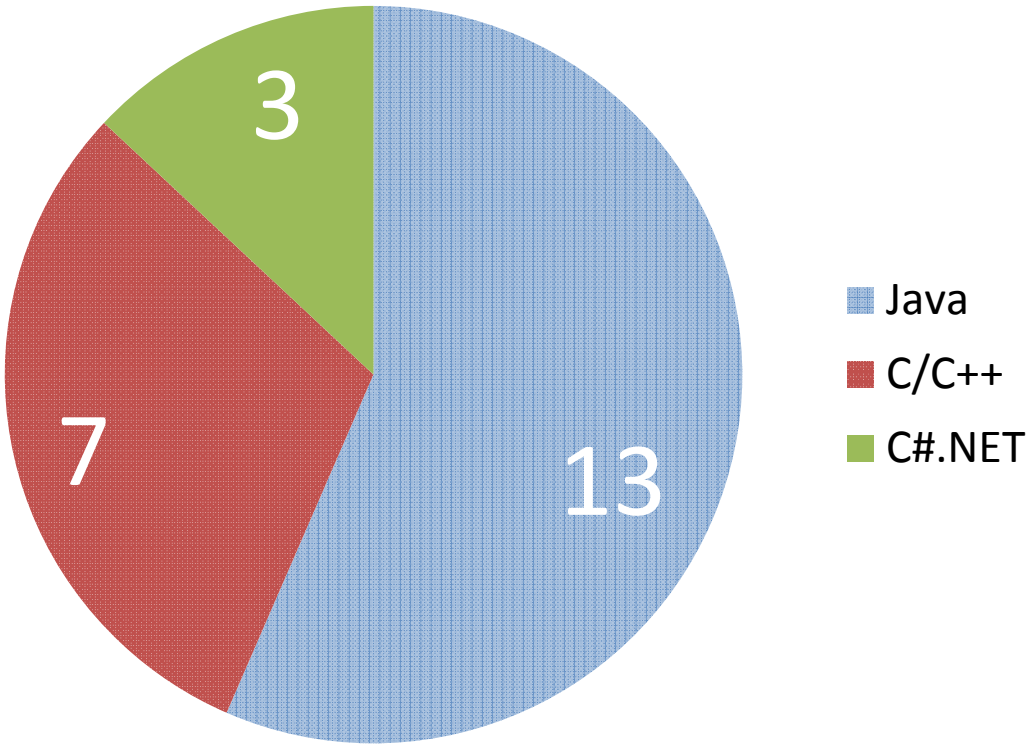
自己紹介【OSS】

- blanco Framework【開発フレームワーク】
- Benten【翻訳支援ツール】
- Eclipse 日本語化【翻訳作業そのもの】
  - 言語パック (サードパーティ版)
  - Pleiades への翻訳投込
- その他多数の OSS を Develop

プログラミング言語経歴

業務における
プログラミング言語の
利用経歴分布
(小学校・中学校時代の
マシン語経験などは除く)

単位: 年



C++経験のほとんどは
Visual C++ によるもの

マイブーム

LLVM / Clang

http://llvm.org/

これも C++Builder に注目する理由のひとつ

モバイルデバイスの普及

モバイルデバイスの普及

- スマートフォン、タブレット端末の急激な普及
- ダウンサイジングや Web 1.0 に匹敵する衝撃

- 複数の OS (iOS と Android のシェアは大きい)
- カメラやセンサ類などの各種デバイス
- タッチ UI
- (PC に比べると) 非力な CPU (あるいはバッテリー容量の制約)
- (やむを得ず) 不安定なネットワーク

- どのようなアプリケーション開発手法が妥当なのか?

モバイルデバイスのアプリケーション開発手法

- 要件
  - マルチ OS、マルチデバイス対応のプログラミング言語
  - CPU コストが少ない言語
    - ネイティブアプリケーション
    - ネットワークの無い環境でも動作し続けること
    - ガベージコレクション (GC) を**搭載しない**のがベター
  - 私は Java および C++ が得意
  - アプリケーションの難読化 (解析困難化) が可能

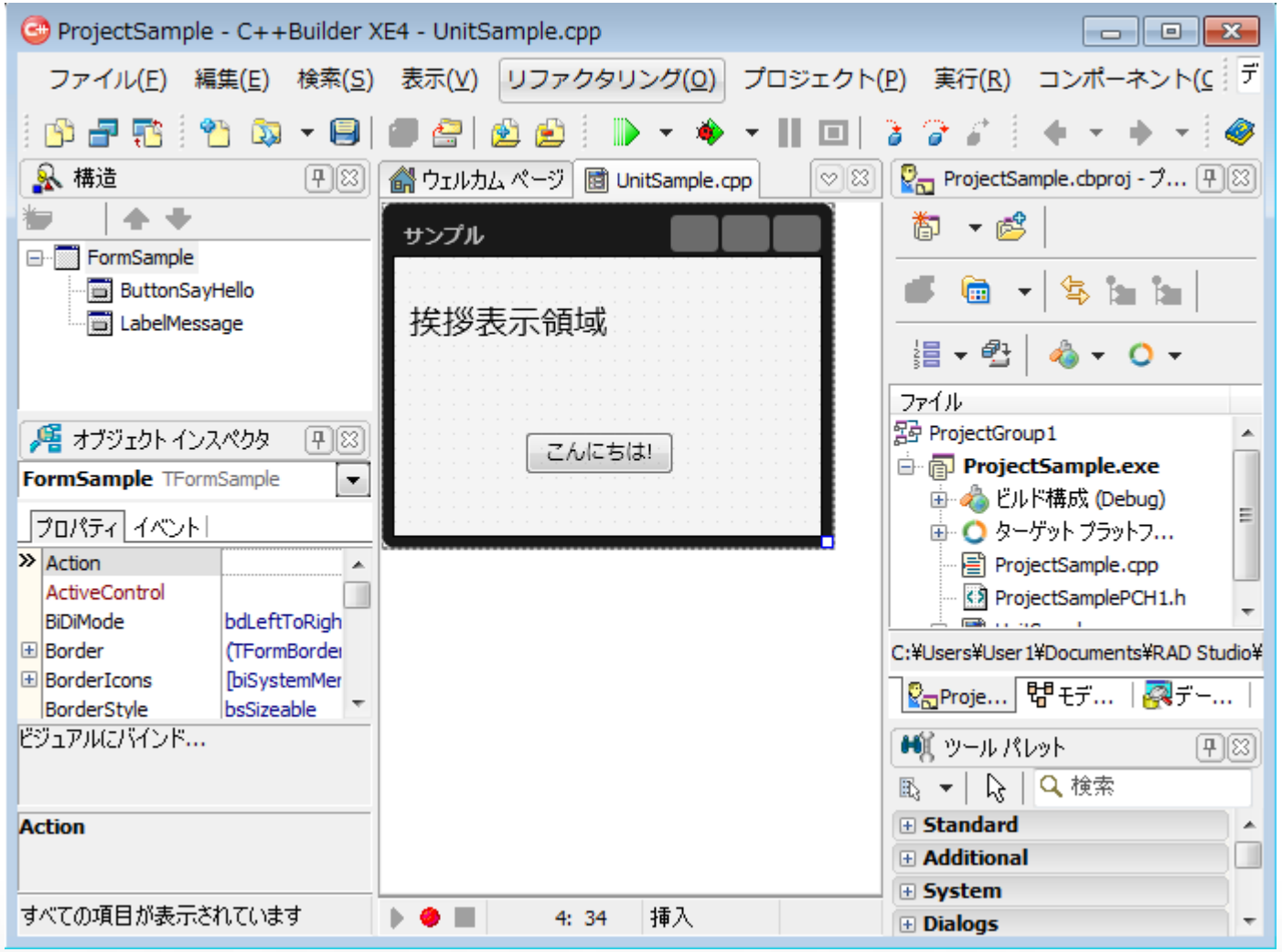
- C++Builder の iOS および Android 対応版 (2013/末に登場?) が要件にフィットする見込み

C++Builder の簡易な紹介

C++Builder の簡易な紹介

- とても長い歴史をもつ C++ 統合開発環境
  - 1997 年 ver.1
  - 2010 年 XE (ver.15)
  - 2013 年 XE4 (ver.18)
  - 参考: Visual C++ 1.0 は 1993 年
- 統合開発環境は Windows 上で動作
- Delphi (Object Pascal) の C++ 版
  - iOS 対応や Android 対応は Delphi が優先される
- 単体で動作できるロードモジュール (EXE) を生成可能
- 2Way ビジュアルデザイナ

C++Builder の画面イメージ



C++Builder のソースコード基本構成

ファイル名		ファイル形式
ProjectSample.cbproj	プロジェクトファイル	XML
ProjectSample.cbproj.local	プロジェクトファイル	XML
ProjectSample.cpp	エントリポイントを含むソースファイル	テキストファイル
ProjectSamplePCH1.h	プリコンパイル済みヘッダファイル	テキストファイル
UnitSample.h	フォームのヘッダファイル	テキストファイル
UnitSample.cpp	フォームのソースファイル	テキストファイル
UnitSample.fmx	フォーム (FireMonkey) のデザイン情報	テキストファイル

ProjectSample.cpp

```
#include <fmx.h>
#pragma hdrstop
#include <tchar.h>
USEFORM("UnitSample.cpp", FormSample);

extern "C" int FMXmain() {
  try {
    Application->Initialize();
    Application->CreateForm(__classid(TFormSample), &FormSample);
    Application->Run();
  }
  catch (Exception &exception) {
    Application->ShowException(&exception);
  }
  catch (...) {
    try {
      throw Exception("");
    }
    catch (Exception &exception) {
      Application->ShowException(&exception);
    }
  }
  return 0;
}
```

比較的シンプルで直観的なコード

FireMonkey アプリケーションのエントリポイント

ProjectSamplePCH1.h

```
#include <fmx.h>
#include <tchar.h>
```

シンプルで直観的なコード

プリコンパイル済みヘッダファイル

UnitSample.h

```
#ifndef UnitSampleH
#define UnitSampleH
#include <System.Classes.hpp>
#include <FMX.Controls.hpp>
#include <FMX.Forms.hpp>
#include <FMX.StdCtrls.hpp>
#include <FMX.Types.hpp>

class TFormSample : public TForm {
__published: // IDE で管理されるコンポーネント
  TButton *ButtonSayHello;
  TLabel *LabelMessage;
  void __fastcall ButtonSayHelloClick(TObject *Sender);

private: // ユーザー宣言

public : // ユーザー宣言
  __fastcall TFormSample(TComponent* Owner);
};

extern PACKAGE TFormSample *FormSample;

#endif
```

シンプルで直観的なコード

フォームのヘッダファイル

UnitSample.cpp

```
#include <fmx.h>
#pragma hdrstop

#include "UnitSample.h"

#pragma package(smart_init)
#pragma resource "*.fmx"
TFormSample *FormSample;

__fastcall TFormSample::TFormSample(TComponent* Owner)
  : TForm(Owner) {
}

void __fastcall TFormSample::ButtonSayHelloClick(
  TObject *Sender) {
  LabelMessage->Text = "ようこそ、こんにちは。";
}
```

シンプルで直観的なコード

フォームのソースファイル

UnitSample.fmx

```
object FormSample: TFormSample
  Left = 0
  Top = 0
  Caption = #12469#12531#12503#12523
  ClientHeight = 152
  ClientWidth = 233
  FormFactor.Width = 320
  FormFactor.Height = 480
  FormFactor.Devices = [dkDesktop, dkiPhone, dkiPad]
  DesignerMobile = False
  DesignerWidth = 0
  DesignerHeight = 0
  DesignerDeviceName = ''
  DesignerOrientation = 0
  object ButtonSayHello: TButton
    Height = 22.000000000000000000
    Position.X = 72.000000000000000000
    Position.Y = 96.000000000000000000
    TabOrder = 0
    Text = #12371#12435#12395#12385#12399'!'
    Width = 80.000000000000000000
    OnClick = ButtonSayHelloClick
  end
  object LabelMessage: TLabel
    Font.Size = 18.000000000000000000
    StyledSettings = [ssFamily, ssStyle, ssFontColor]
    Height = 41.000000000000000000
    Position.X = 8.000000000000000000
    Position.Y = 16.000000000000000000
    Text = #25384#25334#34920#31034#38936#22495
    Width = 217.000000000000000000
  end
end
```

通常は C++Builder のデザイン画面で編集

可読性の良いテキストファイル

フォーム(FireMonkey)のデザイン情報

私にとっての C++Builder

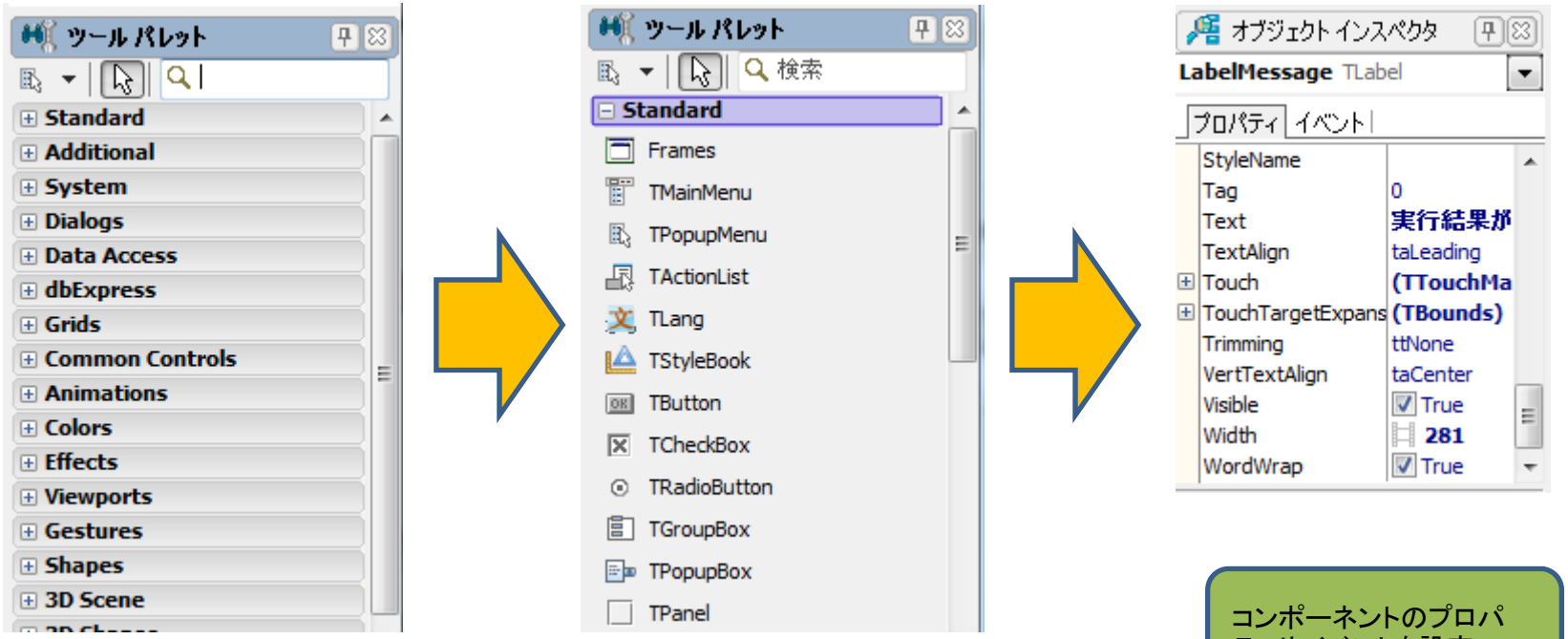
- 商用 Clang ベース統合開発環境
  - ただし XE4 の Windows (32bit) コンパイラは まだ Clang ベースではない…
- iOS および Android 開発環境 (になる予定)
- C++11 学習環境
  - XE4 の Windows (64bit) コンパイラは C++11 対応済み
  - XE4 の Windows (32bit) コンパイラの C++11 対応は微妙…

- 連載記事執筆対象 (苦笑)

コンポーネント (1つめ)

コンポーネントを使う

- ツールパレットからコンポーネントを選んでフォームに貼りつける
  - 任意の場所に配置できる
- オブジェクトインスペクタをもちいてプロパティやイベントを設定する



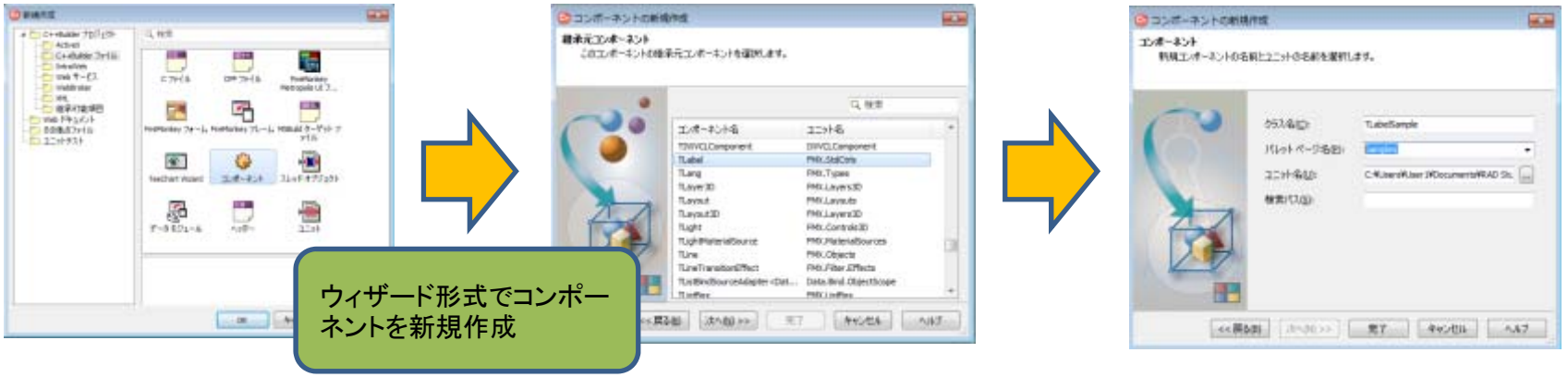
多彩なコンポーネント群から、目的のコンポーネントを選択

フォームに貼りつけ

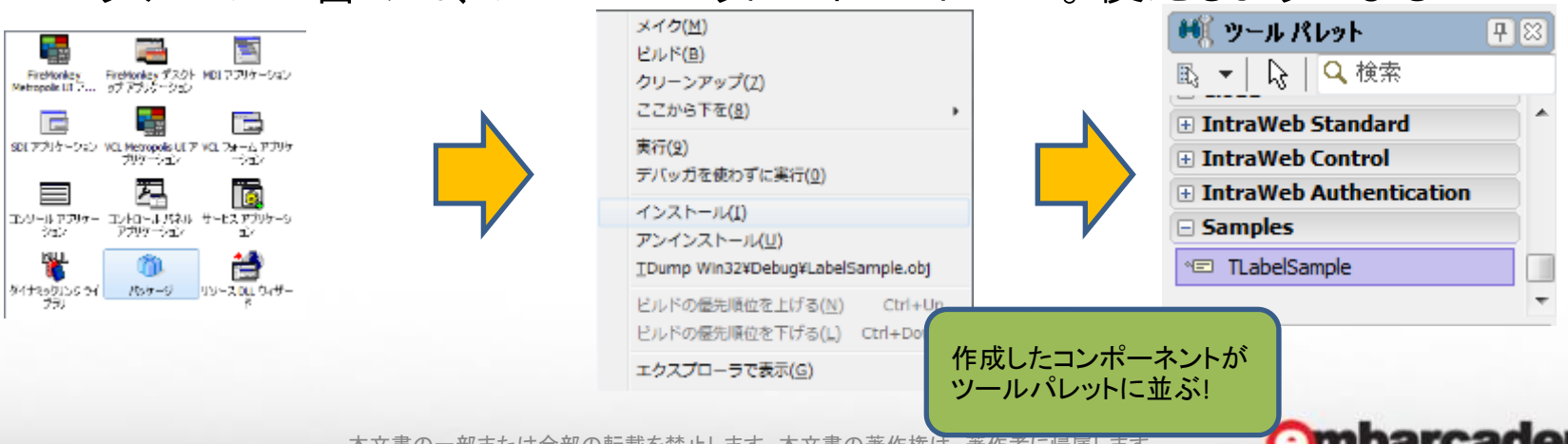
コンポーネントのプロパティやイベントを設定

コンポーネントを作る

- かんたんに画面部品を新規作成可能! (非 GUI 部品も作成可能!)



- パッケージに含めて、ツールパレットにインストール。使えるようになる!



C++Builder のコンポーネント

- とても簡単にコンポーネントが作成できる
  - 積極的なソフトウェア部品の作成および利用が可能
  - 部品化による副作用が少ない
- C++Builder のツールパレットにシームレス統合
- 開発時および実行時の性能的なオーバヘッドが小さい
- ロードモジュール (EXE) に一体化できる
- マルチデバイス対応が可能
- COM / ActiveX コンポーネントとは全く異なるアーキテクチャ
  - モジュール配布時の登録作業(regsvr32.exe 的なもの)などは不要

言語の機能 (2つめ)

言語の機能

- C++11 対応 (あるいは TR1)
- C++ 標準ライブラリ (あるいは Boost)
- LLVM / Clang

- C++Builder の Clang 対応状況
  - Windows (64bit) コンパイラは XE4 において Clang ベース
  - Windows (32bit) コンパイラおよび Mac OS X コンパイラは XE4 において 非 Clang ベース
    →将来のバージョンにおいて Clang ベースになるのかな?
  - iOS や Android 対応は、おそらく Clang ベースで実現される模様

C++11

(ISO/IEC 14882:2011)

- C++11 がもたらす C++ ルネッサンス
  - C++11 (ISO/IEC 14882:2011) における驚異のパワーアップ
  - 劇的な進化
  - C++11 以降において、C++ は再評価されるべき

- 言語仕様の充実
  - スレッドが言語仕様に含まれました! (含む mutex)
  - メモリリークを発生させなくするための仕組み (unique_ptr、shared_ptr) が導入されました! new – delete の手動記述は控えることができる。(循環参照に注意)
- C++ 標準ライブラリの充実
  - コンテナ関連 API の整備
  - ありがとう Boost!

LLVM / Clang

- LLVM
  - コンパイラ基盤
  - モジュール式で再利用可能なコンパイラとツールチェーン技術のコレクション
- Clang
  - LLVM コンパイラ用の C、C++、Objective C、Objective C++ フロントエンド

マルチデバイスのサポート (3つめ)

マルチデバイスのサポート

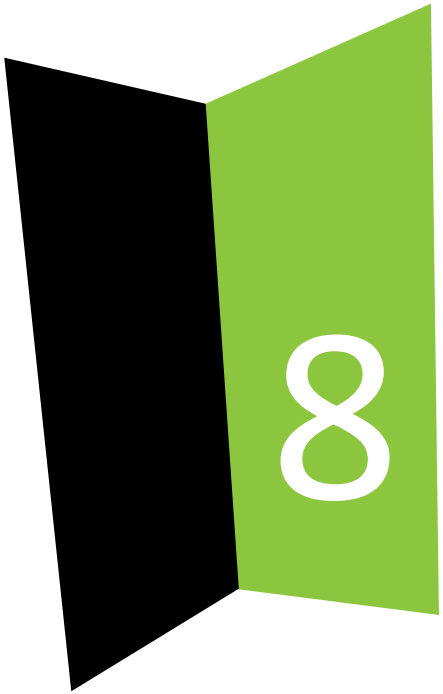
- FireMonkey でマルチデバイス対応を実現
  - ※FireMonkey の説明は次ページ
- LLVM / Clang でマルチデバイス対応を実現予定
  - コンパイラのマルチデバイス対応
  - Windows、Mac OS X、iOS、Android が対応される予定

- 単一プロジェクトから Windows (32bit, 64bit)、Mac OS X に加えて iOS、Android のネイティブアプリケーション生成が可能になることを期待。

FireMonkey

- ユーザインタフェースのマルチデバイス対応
  (デバイスの抽象化を実現するライブラリ)
- Java Swing とアーキテクチャ的に似ている模様

- FireMonkey ライブラリのファイルサイズに注意



まとめ

まとめ (1)

- モバイルデバイス向けのアプリケーション開発手法についての「解」を準備すべきタイミングなのでは?
  - モバイルデバイスから組込系の風味が感じられる

- C++Builder の実現方式(予定)は「スジ」が非常によさそうに見える
  ★LLVM / Clang + FireMonkey + C++11

- C++11 により C++ 言語は再評価されるべき

まとめ (2)

- C++Builder の iOS + Android 対応版が計画通りにリリースされれば、モバイルアプリケーション開発における重要な選択肢となる可能性を予測
  - 計画通りにリリースされることを祈ります。

参考情報

- 個人的な推測による参考情報
  - C++Builder のみ利用する場合であっても、モバイルアプリケーション開発をともなう用途を目的にプロダクトを購入する場合には、商品としては RAD Studio (Professional 版またはそれ以上) を購入したうえで、さらにサポート・メンテナンス・プログラムを契約するような組み合わせを選択することが必要になる可能性が高いのではないのか、と予想しています…

Q&A