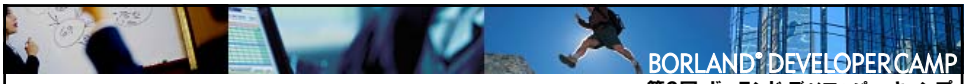
チュートリアルセッション#2

Java, Delphi, C++Builderユーザのための メモリリーク, ボトルネックの検出手順

Borland®



講師紹介

- 高橋智宏
- 1973年生まれ、京都大学 法学部卒
- エバンジェリスト 兼 コンサルタント 兼 トレーナー…
- 学生の時購入したTurboC++2ndからの熱狂的なボーランドファン
- 参加しているメーリングリストやコミュニティ
  - JBuilder ML,C++Builder ML,Delphi ML,C# ML,CORBA ML 等…
  - ミクシィ
    - http://mixi.jp/show_friend.pl?id=208738
- 「Java読書会」を運営
  - http://www.javareading.com/bof/

Borland®

アジェンダ

- Javaのメモリリークの落とし穴
  - OptimizeIt Profiler for Java
- C++Builder6,C++Builder2006のメモリリークの落とし穴
  - CodeGuard
- Delphi for Win32のメモリリークの検出方法
  - MemCheck, FastMM(BDS2006), 本家FastMM
- Delphi for .NETのメモリリークの落とし穴
  - OptimzieIt Profiler for .NET1.1
- CPUプロファイリング手順
  - Java, C++Builder, Delphi for Win32, Delphi for .NET
  - Delphi7 と BDS2006 でメモリマネージャの比較

BORLAND® DEVELOPER CAMP
第2回 ボーランド デベロッパー キャンプ

Javaのメモリリークの落とし穴

- 基本的にJavaにはメモリリークが無いハズ…

```
public class Frame1 extends JFrame {
  JButton jButton1 = new JButton();
  public Frame1() {
    ...
    jbInit();
  }
  private void jbInit() throws Exception {
    ...
    jButton1.setText("フレームを表示");
    jButton1.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jButton1_actionPerformed(e);
      }
    });
  }

  // フレーム(test.Frame2)を表示
  public void jButton1_actionPerformed(ActionEvent e) {
    test.Frame2 f = new test.Frame2(); // Frame2を生成
    f.setVisible(true);                // Frame2を表示
  }
}
```

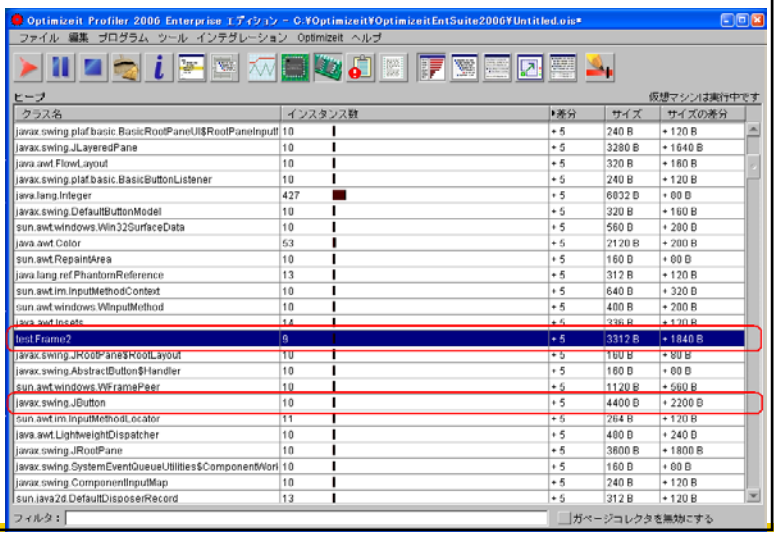
```
public class Frame2 extends JFrame {
  JButton jButton1 = new JButton();
  public Frame2() {
    ...
    jbInit();
  }
  private void jbInit() throws Exception {
    ...
    jButton1.setText("閉じる");
    jButton1.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jButton1_actionPerformed(e);
      }
    });
  }

  public void jButton1_actionPerformed(ActionEvent e) {
    setVisible(false); // Frame2(自分自身)を閉じる
  }
}
```

Borland®
Copyright (C) 2006, Borland Software Corporation. 本文書の一部または全部の転載を禁止します。

OptimizeIt Profiler for Java で監視してみる

- 何故か test.Frame2 を含む様々なインスタンスがガベコレされない…

メモリプロファイラで見るとインスタンス数が増えたまま



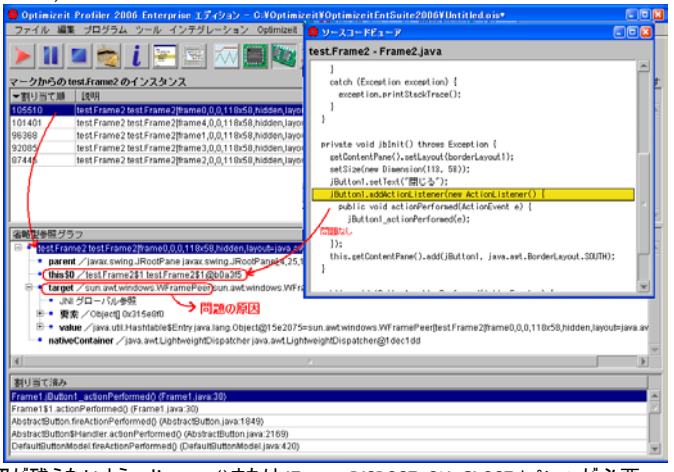
Borland®

BORLAND® DEVELOPER CAMP
第2回 ボーランド デベロッパー キャンプ

test.Frame2インスタンスは確かに参照されている…

- 自分のコード(イベントハンドラ)が参照しているのは当たり前だが…

参照ツリーを使って、
参照されている様子を確認



- ネイティブピアからの参照が残らないよう、dispose()またはJFrame.DISPOSE_ON_CLOSEオプションが必要

Borland®

C++Builder特有の落とし穴

- try/__finally でメモリリークを防ぐ

```
class MyClass {
};

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  MyClass* p = NULL;
  try {
    p = new MyClass();
    try {
      int x = StrToInt("xyz");
    }
    catch(const Exception& ex) {
      return;
    }
    catch(...) {
      return;
    }
  }
  __finally {
    if(p) {
      delete p; // 解放!!
    }
  }
}
```

- VCLインスタンスのリーク

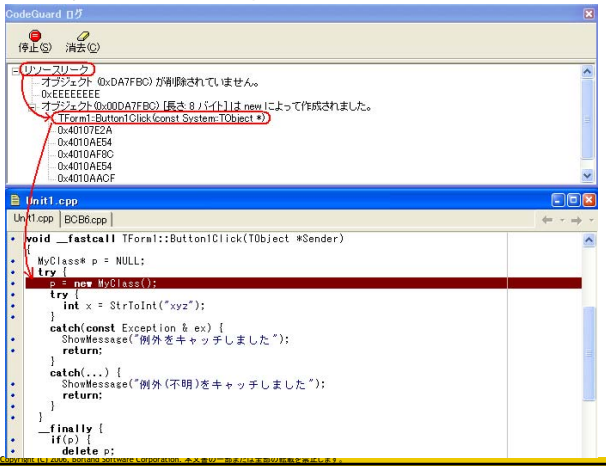
```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  new TButton((TComponent*)NULL); // Owner無し!!
}
```


BORLAND® DEVELOPER CAMP
第2回 ボーランド デベロッパー キャンプ

C++Builder6 の CodeGuard で確認してみる

- C++Builder6では
  - なぜか __finally が呼び出されない（仕様？）
  - TButtonインスタンスのリークは見つけられない（仕様？）

MyClassオブジェクトが削除されていないと報告されました。

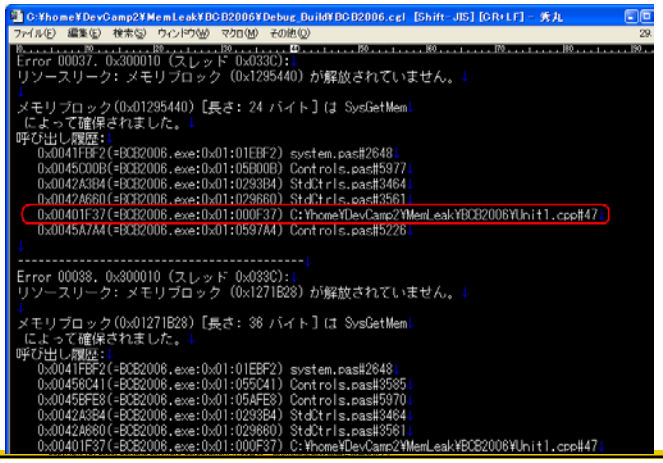


- オススメは
  - std::auto_prt
  - boost::scoped_ptr

C++Builder2006では？

- catchの中からreturnしても __finallyが呼び出されるようになりました
- VCLのインスタンスのリークも報告されるようになりました
  - 但し、動的RTL(cc3270xx.dll)とパッケージ(xxxx.bpl)はOFFにする

CodeGuardのログファイルに、
ソースコードと行番号が出力されています



Borland®

BORLAND® DEVELOPER CAMP
第2回 ボーランド デベロッパー キャンプ

C++Builder と catch(…)

- 例外の補足に catch(…) のみを使用するのってどう？
- VCL例外に関連するメモリがリークするので注意!! (バグ？)
  - C++Builder2006のCodeGuardでチェックしてみよう

誤

```
try {
  int x = StrToInt("xyz");
}
catch(…) {
  ShowMessage("…");
}
```

正

```
try {
  int x = StrToInt("xyz");
}
catch(const Exception& ex) {
  ShowMessage("…");
}
catch(…) {
  ShowMessage("…");
}
```

Borland®

Delphi5～Delphi2005(Win32) - MemCheck

- Delphi2005までは MemCheck がオススメ
  - http://v.mahon.free.fr/pro/freeware/memcheck/
- 無料(MemCheck.pasのみ)
- Delphi2006では利用不可！！
  - Delphiコンパイラやメモリマネージャの変更による影響の為

Borland

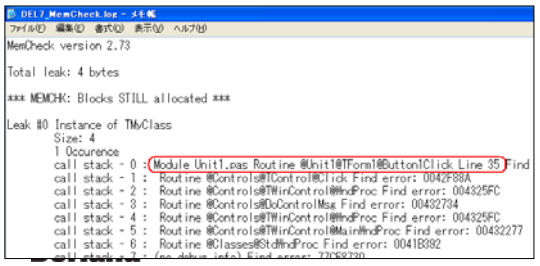
BORLAND® DEVELOPER CAMP
第2回 ボーランド デベロッパー キャンプ

Delphi5～Delphi2005(Win32) - MemCheck (続き)

- MemCheck.pasをプロジェクトに追加
- MemChk; の呼び出しを追加
- 「スタックフレームの生成」をON
- 「TD32デバッグ情報を含める」をON
- プログラム終了時にログファイルが生成される

```
begin
  MemChk; // チェック開始
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

```
type
  TMyClass = class
  end;

procedure TForm1.Button1Click(Sender: TObject);
var
  p: TMyClass;
begin
  p := TMyClass.Create; // 破棄されていない
end;
```

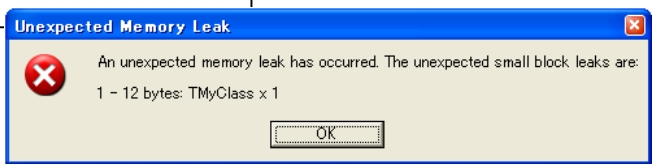


Copyright (C) 2006, Borland Software Corporation. 本文書の一部または全部の転載を禁止します。

Delphi2006 for Win32 – FastMM

- BDS2006から、メモリマネージャがFastMMベースに置き換わった
  - BDNの記事 http://bdn.borland.com/article/33624
  - 今までより高速。後ほど確認します
- メモリリーク報告機能が付属している
  - ReportMemoryLeaksOnShutdown := True; の行を追加するだけ
  - プログラム終了時にダイアログボックスが表示される
  - しかし、インスタンスの生成場所までは教えてくれない！

```
begin
  ReportMemoryLeaksOnShutdown := True; // これだけ
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```



Borland

BORLAND® DEVELOPER CAMP
第2回 ボーランド デベロッパー キャンプ

Delphi2006 with 本家FastMM

- リークしたインスタンスの生成場所は、実は Delphi2006 でも調べることができる!!
  - BDS2006付属のFastMMを、本家FastMMの最新版に置き換えよう
  - http://sourceforge.net/project/showfiles.php?group_id=130631
  - ライセンスはMPL
  - 最新版は4.70 (※)

(※)Delphi2006以前のバージョンでも
利用可能

Borland

Delphi2006 with 本家FastMM (続き)

- usesに ShareMemユニット を追加して、borlndmm.dll(共用メモリマネージャ) がロードされるように変更する
- 本家FastMMの「Debug版 BorlndMM.dll」がロードされるようにする
- 本家FastMMの「FastMM_FullDebugMode.dll」が実行時に必要
- 「スタックフレームの生成」をON
- 「TD32デバッグ情報を含める」をON
- BDS2006のIDEが起動している必要あり
- プログラム終了時にダイアログボックスが表示される
- プログラム終了時にログファイルが生成される
  - borlndmm_MemoryManager_EventLog.txt
- プログラムは最終的にAVで停止することがあるが「気にせず」タスクマネージャやIDE上で無理矢理終了させる

Borland

BORLAND® DEVELOPER CAMP
第2回 ボーランド デベロッパー キャンプ

Delphi2006 with 本家FastMM (続き)

```
program XXXX;

uses
  ShareMem, // これだけ
  Forms,
  Unit1 in 'Unit1.pas' {Form1},
...
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```


```
--- borlndmm_MemoryManager_EventLog.txt ----------------------
A memory block has been leaked. The size is: 4

Stack trace of when this block was allocated (return addresses):
402C9A [System][@GetMem]
40383B [System][TObject.NewInstance]
403BAA [System][@ClassCreate]
403870 [System][TObject.Create]
403A33 [System][@IsClass]
453693 [Unit1.pas][Unit1][TForm1.Button1Click][34]
437052 [Controls][TControl.Click]
426537 [StdCtrls][TButton.Click]
426635 [StdCtrls][TButton.CNCommand]

The block is currently used for an object of class: TMyClass
...
...
```

Borland

Copyright (C) 2006, Borland Software Corporation. 本文書の一部または全部の転載を禁止します。

Delphi for .NET特有の落とし穴

- 基本的に.NETにはメモリリークが無いハズ…

```
unit Unit1;
…
type
  TForm1 = class(TForm)
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
    …
  end;
…
…
implementation

uses Unit2;

procedure TForm1.Button1Click(Sender: TObject);
var
  f: TForm2;
begin
  f := TForm2.Create(nil); // TForm2を生成(Ownerはnil)
  f.ShowModal;             // TForm2を表示
end;
```

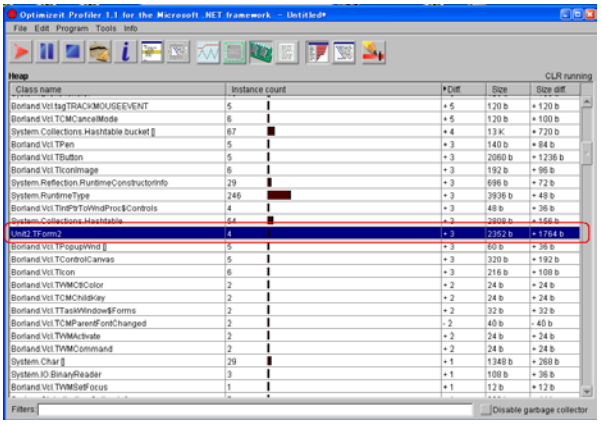
```
unit Unit2;
…
type
  TForm2 = class(TForm)
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
    …
  end;
…
…
implementation

procedure TForm2.Button1Click(Sender: TObject);
begin
  Close; // TForm2(自分自身)を閉じる
end;

end.
```

Borland

BORLAND® DEVELOPER CAMP
第2回 ボーランド デベロッパー キャンプ

OptimizeIt Profiler for .NET1.1(※) で監視してみる

- 何故か Unit2.TForm2 等のインスタンスがガベコレされない…



- Ownerがnilでも、デフォルトのOwnerがいるので、x.Free;またはFreeAndNil(x);の呼び出しは必須

Borland

(※)単体販売は終了しています。Delphi2005 Architect版にのみライセンスが付属しています。

Copyright (C) 2006, Borland Software Corporation. 本文書の一部または全部の転載を禁止します。

CPUプロファイリング - Java

```
public class Frame1 extends JFrame {
  ...
  JButton jButton1 = new JButton();
  public Frame1() {
    jbInit();
  }

  private void jbInit() throws Exception {
    ...
    jButton1.setText("jButton1");
    jButton1.addActionListener(new Frame1_jButton1_actionAdapter(this));
    contentPane.add(jButton1, java.awt.BorderLayout.SOUTH);
  }

  public void jButton1_actionPerformed(ActionEvent e) {
    long st = System.currentTimeMillis();
    long et = st;
    while( (et-st) < 5000 ) {
      Integer.parseInt("123");
      et = System.currentTimeMillis();
    }
    JOptionPane.showMessageDialog(this, "終わりました");
  }
}
...
```

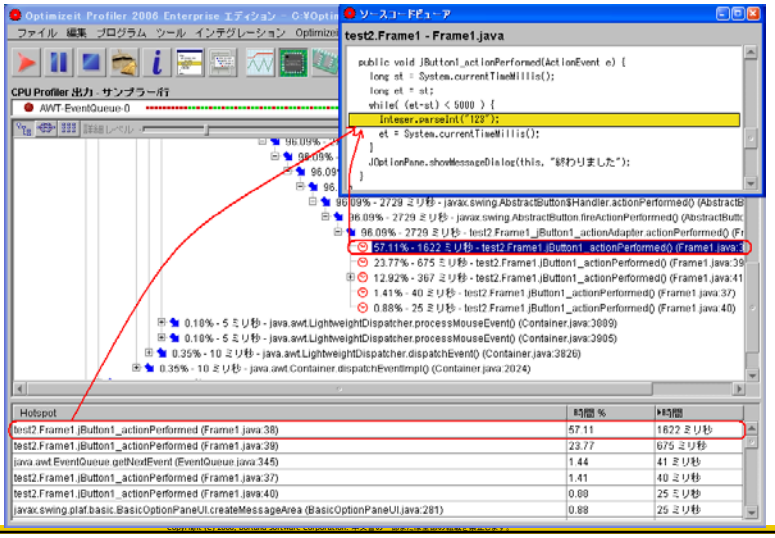
5秒間のループ処理

Borland

BORLAND® DEVELOPER CAMP
第2回 ボーランド デベロッパー キャンプ

OptimizeIt Profiler for Java によるCPUプロファイリング

- デフォルトは、5ミリ秒間隔のサンプリングを実施



Borland

CPUプロファイリング - C++Builder

```
...
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  unsigned long s = GetTickCount();
  unsigned long e = s;
  while( (e-s) < 5000 )
  {
    char buf[4];
    sprintf(buf, "%s", "123");
    e = GetTickCount();
  }
  ShowMessage("終わりました");
}
...
```

5秒間のループ処理

Borland

BORLAND® DEVELOPER CAMP
第2回 ボーランド デベロッパー キャンプ

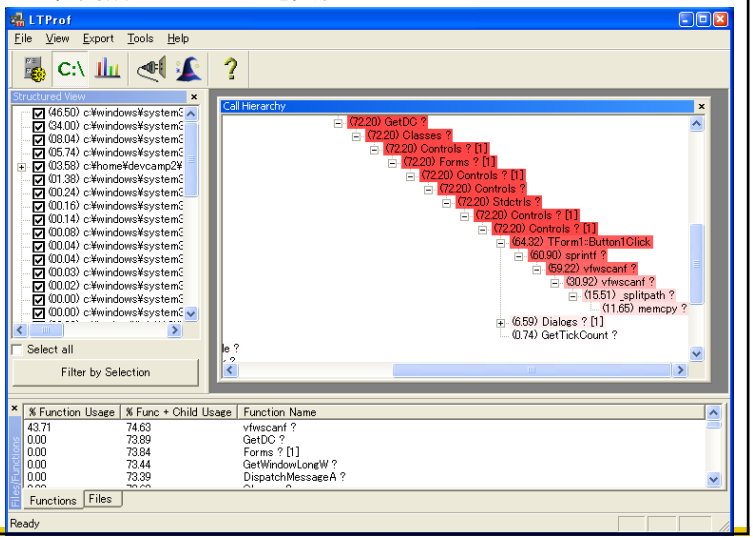
CPUプロファイリング - C++Builder (続き)

- LTProf の紹介
  - Lightweight Technologies社の製品
  - http://www.lw-tech.com/
  - C++Builder/Delphiに対応したCPUプロファイラ
  - 価格は $49.95(USD)
  - TD32デバッグ情報だけでOK

Borland

LTProfによるCPUプロファイリング

- デフォルトは、3ミリ秒間隔のサンプリングを実施



処理時間が長いメソッドが赤くハイライトされています

Borland

BORLAND® DEVELOPER CAMP
第2回 ボーランド デベロッパー キャンプ

CPUプロファイリング - Delphi for Win32

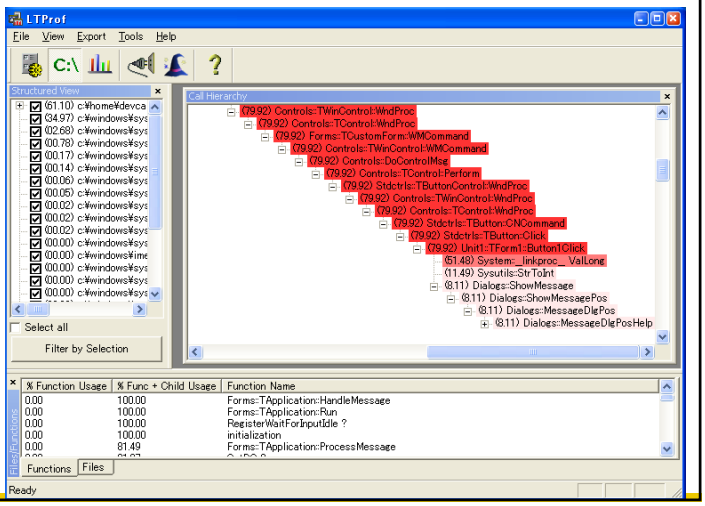
```
...
procedure TForm1.Button1Click(Sender: TObject);
var
  s,e: Cardinal;
  i: Integer;
begin
  s := GetTickCount;
  e := s;
  while (e-s) < 5000 do
  begin
    i := StrToInt('123');
    e := GetTickCount;
  end;
  ShowMessage('終わりました');
end;
...
```

5秒間のループ処理

Borland

LTProfによるCPUプロファイリング

- 「TD32デバッグ情報を含める」をON
- 「デバッグ版DCUを使う」をON



処理時間が長いメソッドが
赤くハイライトされています

Borland

CPUプロファイリング - Delphi for .NET

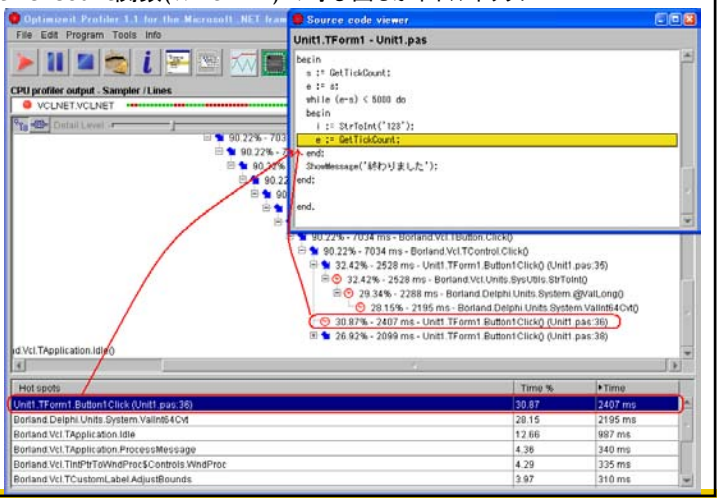
```
...
procedure TForm1.Button1Click(Sender: TObject);
var
  s,e: Cardinal;
  i: Integer;
begin
  s := GetTickCount;
  e := s;
  while (e-s) < 5000 do
  begin
    i := StrToInt('123');
    e := GetTickCount;
  end;
  ShowMessage('終わりました');
end;
...
```

5秒間のループ処理

Borland

OptimizeIt Profiler for .NET1.1 によるCPUプロファイリング

- デフォルトは、5ミリ秒間隔のサンプリングを実施
- 意外にも(？) GetTickCount関数(Win32API) の呼び出しがボトルネックに



処理時間の長い行がハイライトされています

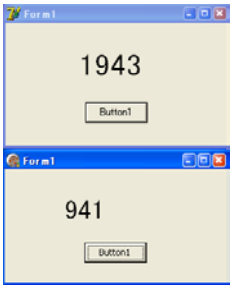
Borland

Delphi7 と BDS2006 でメモリマネージャの比較

- Delphi7は、従来のメモリマネージャを使用
- BDS2006は、製品に付属するFastMMベースのものを使用

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i,j:  Integer;
  s,e:  Cardinal;
  list: TStringList;
begin
  s := GetTickCount;
  for i := 1 to 100 do
  begin
    list := TStringList.Create;
    for j := 1 to 10000 do
    begin
      list.Add(IntToStr(J));
    end;
    FreeAndNil(list);
  end;
  e := GetTickCount;

  Label1.Caption := IntToStr(e-s);
end;
```

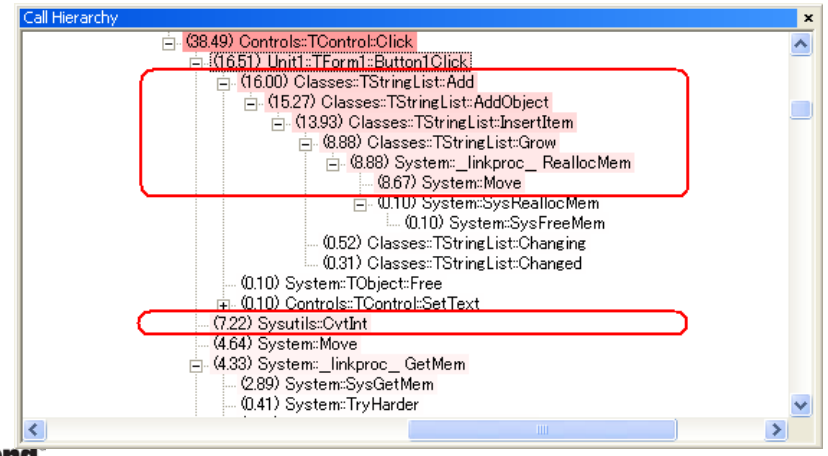
TStringListを使った重いループ処理



Borland

Delphi7だと...

- 大量のTStringList.Addに伴うメモリの再配置処理(ReallocMem,Move)が最も重く
- IntToStrメソッド(Sysutils::CvtInt)がその次に重い

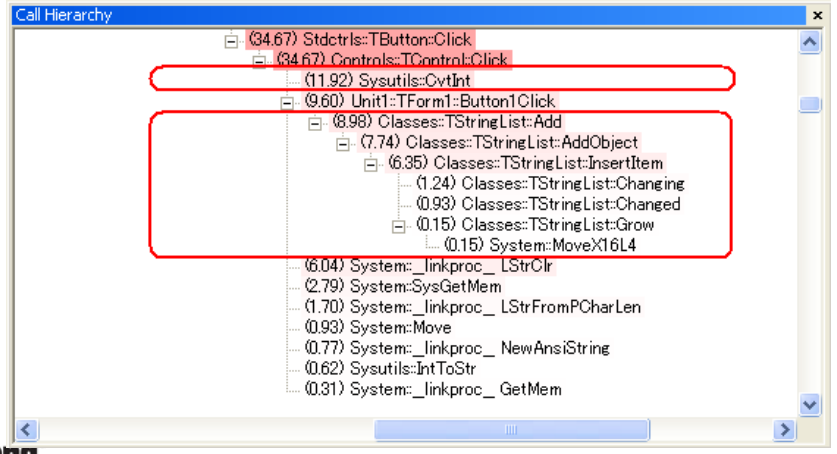


Borland

BORLAND® DEVELOPER CAMP
第2回 ボーランド デベロッパー キャンプ

BDS2006だと...

- メモリの再配置処理は軽くなり、逆にIntToStr(Sysutils::CvtInt)が目立つ結果に。
  - Sysutils::CvtIntの実装内容は、Delphi7とBDS2006とで同一です



Borland

Copyright (C) 2006, Borland Software Corporation. 本文書の一部または全部の転載を禁止します。

Q&A

Borland®