embarcadero
Developer Camp

【T2】テクニカルセッション

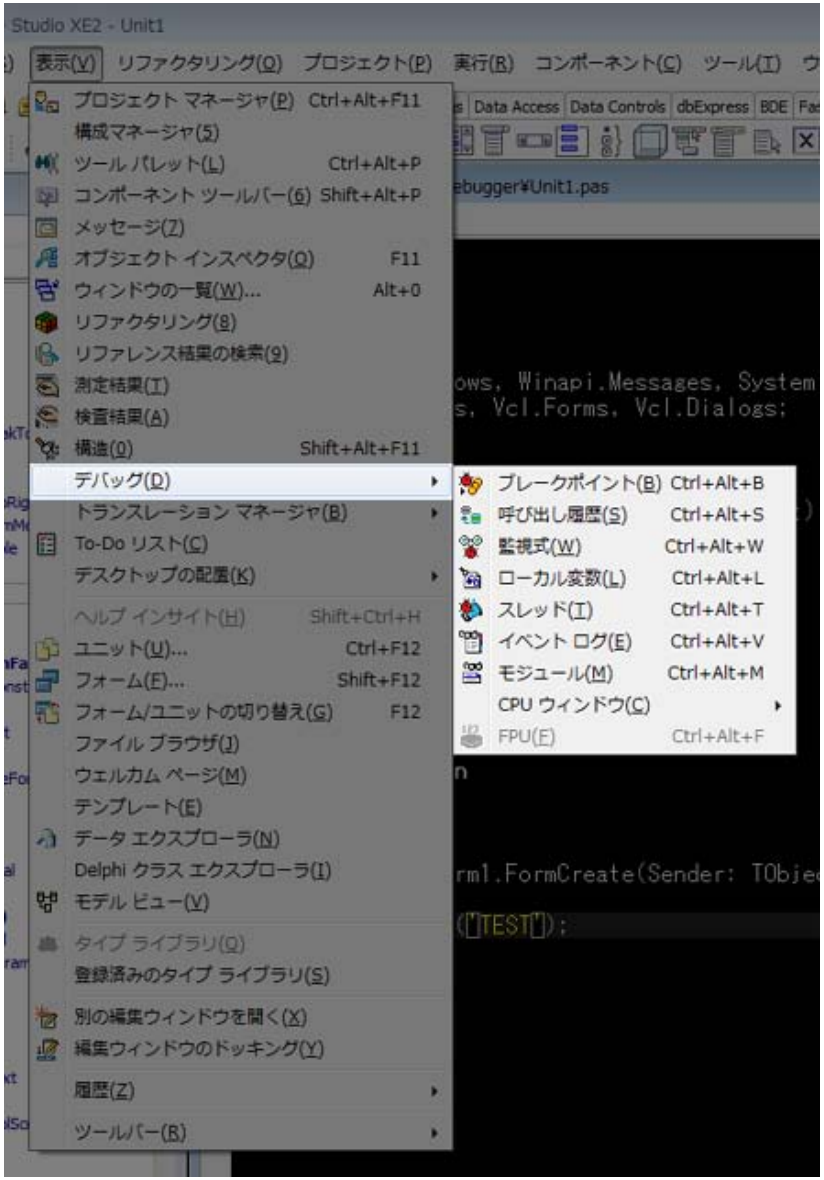
「実践！Delphiデバッグテクニック」

株式会社シリアルゲームズ
取締役
細川　淳

embarcadero
Developer Camp

IDE統合デバッガ

Delphi の IDE 統合デバッガ



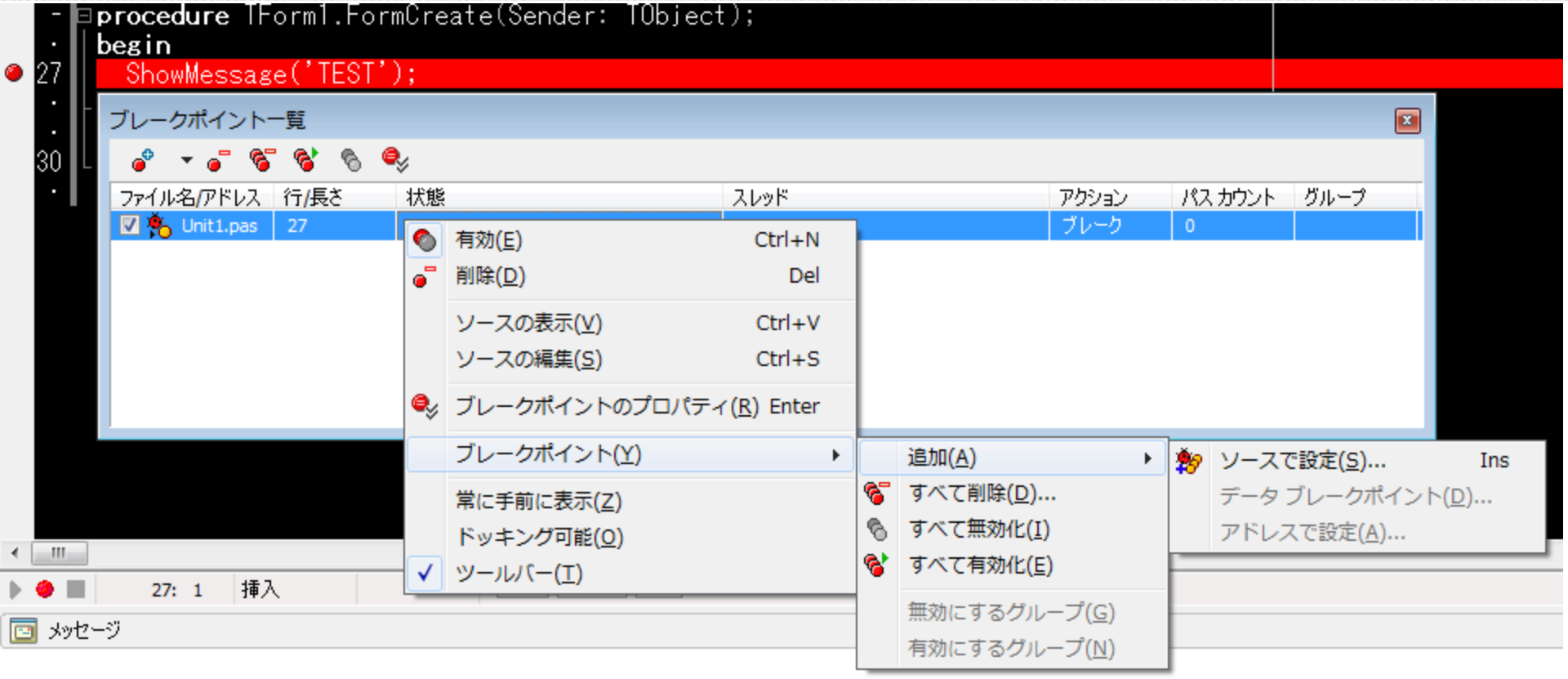
- IDE 統合デバッガについて

「表示」→「デバッグ」で表示される項目ごとに見ていきます

embarcadero
Developer Camp

ブレークポイント一覧

ブレークポイント一覧



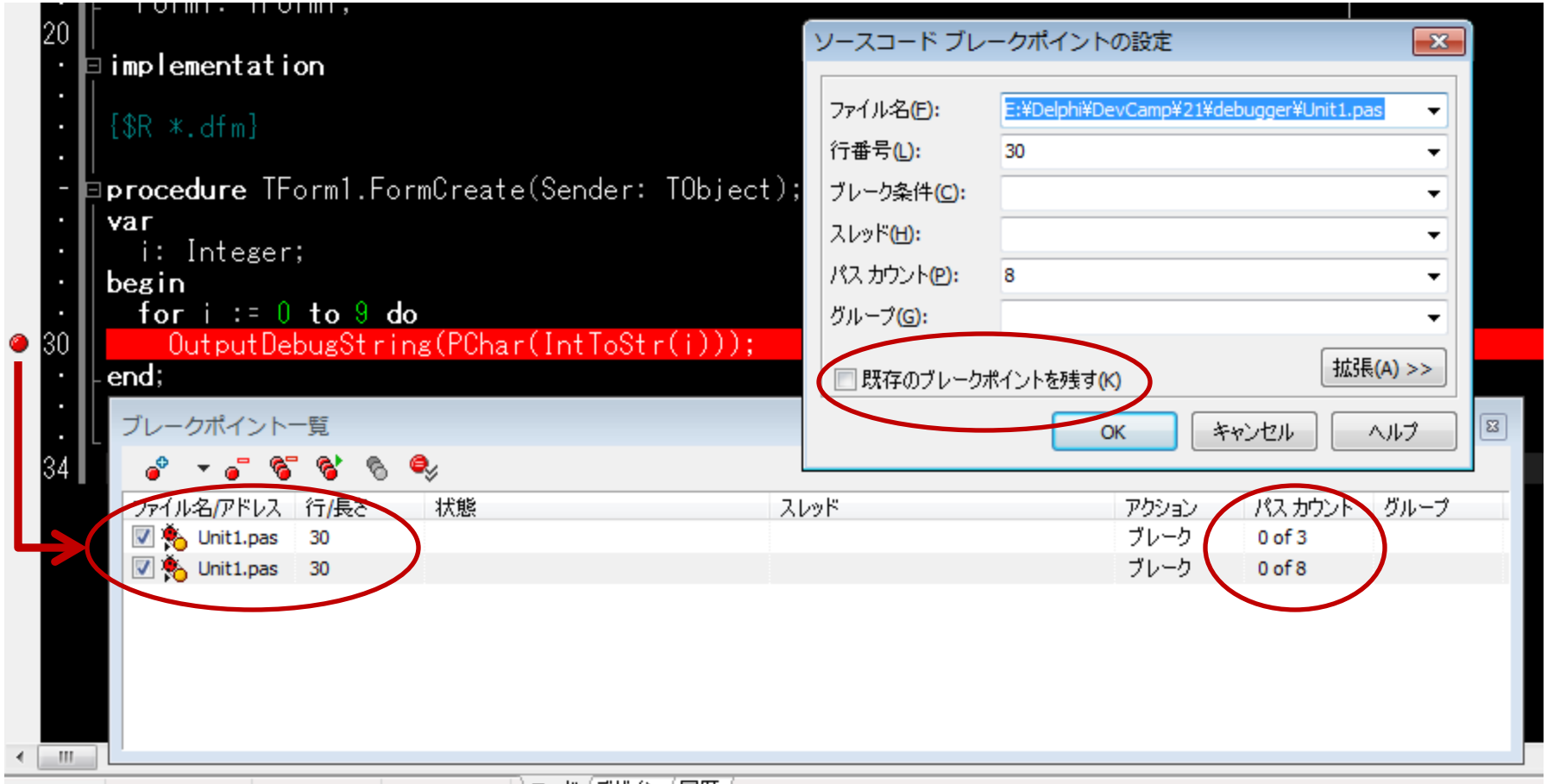
ブレークポイント一覧では、一覧以外にも、プロパティや有効無効、データブレークポイント、アドレスで設定、などなど豊富な機能が利用できます。
ここでは特に「プロパティ」と「データブレークポイント」「アドレスで設定」を見ていきます。

ブレークポイント一覧ープロパティ(条件)

- ブレーク条件とパスカウント
  - プロパティを開くと「ブレーク条件」と「パスカウント」という項目があります。
  - ブレーク条件
    - if 文の条件式と同じように条件を書けます
      - たとえば「A <> 0」などのように条件を書けます
  - パスカウント
    - そのブレークポイントを何回通った時に停止するかを指定できます。
      - たとえばパスカウントに 3 を指定すると3回目にブレークポイントを通った時に停止します。
- この2つの条件を上手く使うとデバッグが捗ります

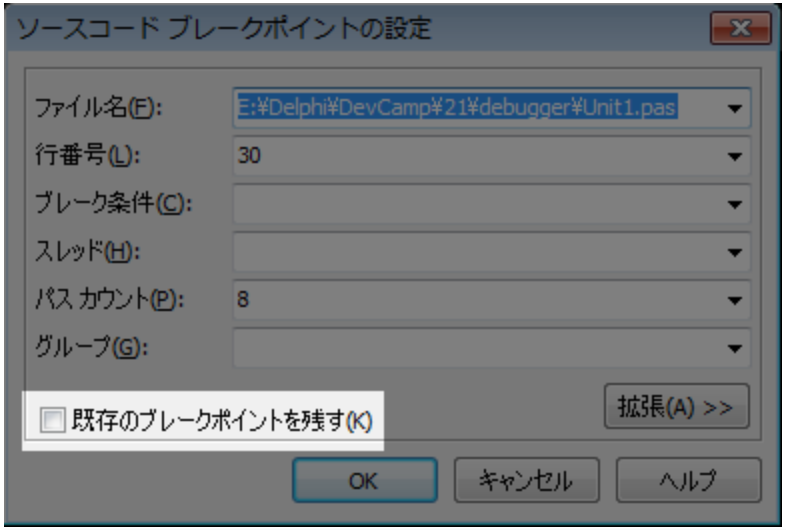
ブレークポイント一覧ー複数の条件

- ブレークポイントで特筆すべきはもう1つ
  - 同じ行に違うプロパティを持つ複数のブレークポイントを持てます。

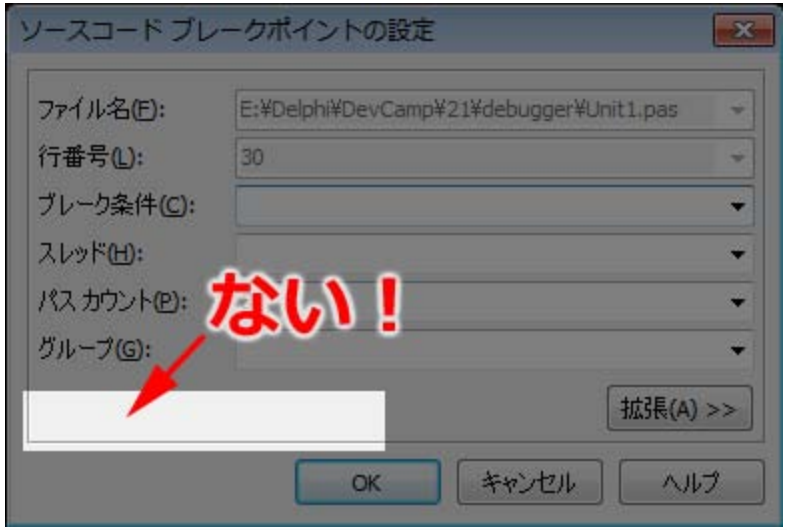


ブレークポイント一覧ー複数の条件

- 複数の条件を持たせるには「既存のブレークポイントを残す」を利用します。
  - これによって、違う条件を持った複数のブレークポイントを設定できます
    - 注意したいのは、これは「ブレークポイント一覧」ウィンドウからプロパティを表示した時にしか現れないチェックです



ブレークポイント一覧から開いたプロパティ



ブレークポイントのコンテキストメニューから開いたプロパティ

ブレークポイント一覧ーブレークポイントの種類

- ブレークポイント一覧では次の3つのブレークポイントを設定できます

名前	概要	設定可能な状態
ソースブレークポイント	ソースコード上の任意の行に設定できるブレークポイントです	通常時・実行時
アドレスブレークポイント	アドレスに対して設定できるブレークポイントです	実行時
データブレークポイント	変数などのデータが変更される時にブレークするブレークポイントです	実行時

- アドレスブレークポイント

  アドレスブレークポイントはユーザーにプログラムを納品した後に威力を発揮します。

  例外発生ウィンドウにはアドレスが出ています。

  ユーザーから冷害発生時のアドレスを聞いて、そこで停止するようにすれば、例外の原因を追うことができます。

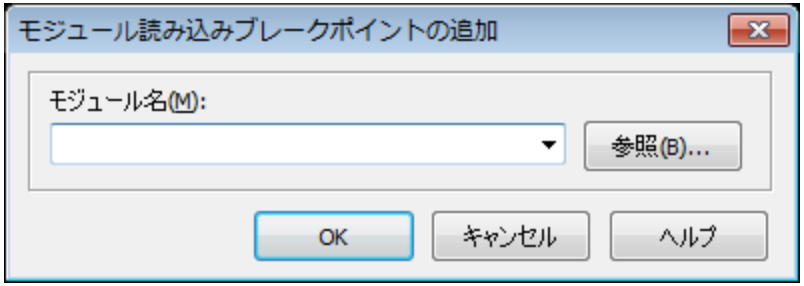
- データブレークポイント

  データブレークポイントはデータ（変数）の変更時に停止します。

  そのため、どこでデータが変更されたのか？を追うときに役に立ちます。

ブレークポイント一覧ーその他のブレークポイント

- モジュールロードブレークポイント
  - ブレークポイント一覧からは設定できず「実行」→「ブレークポイントの追加」→「モジュールロードブレークポイント」から設定できます。
  - DLL など他のモジュールを使っている場合に、モジュールがロードされると停止できます。



embarcadero
Developer Camp

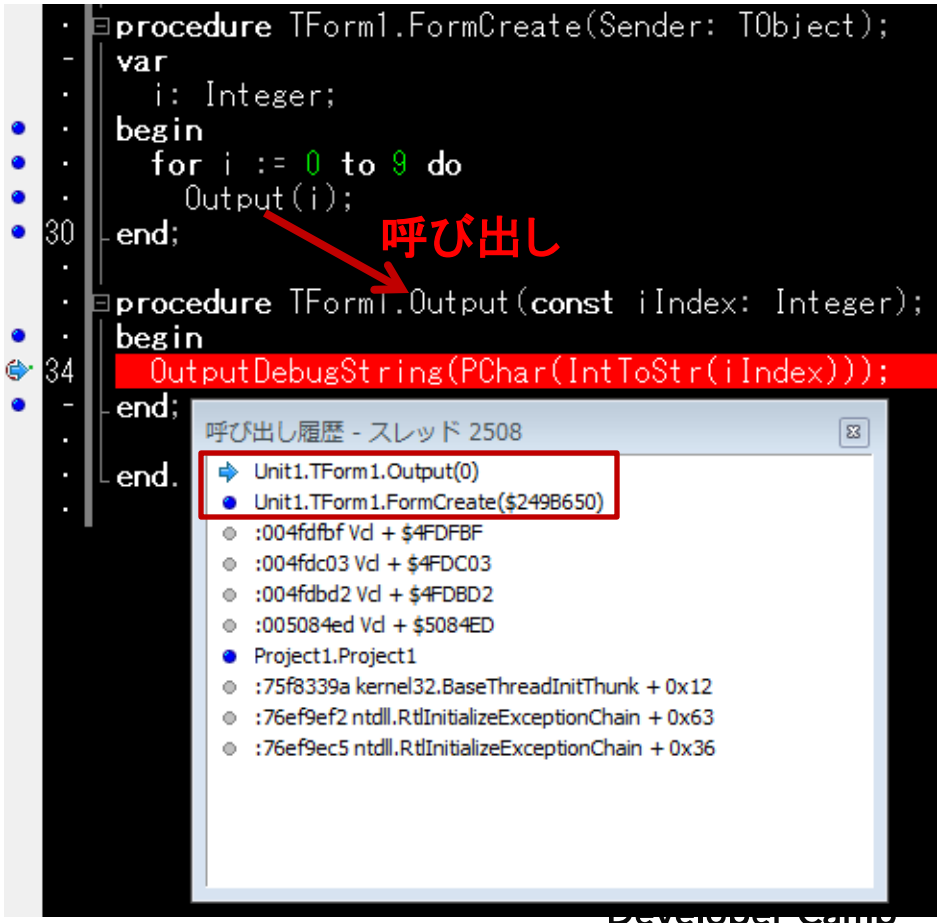
ブレークポイントの
デモンストレーション

embarcadero
Developer Camp

呼び出し履歴と
ローカル変数

呼び出し履歴

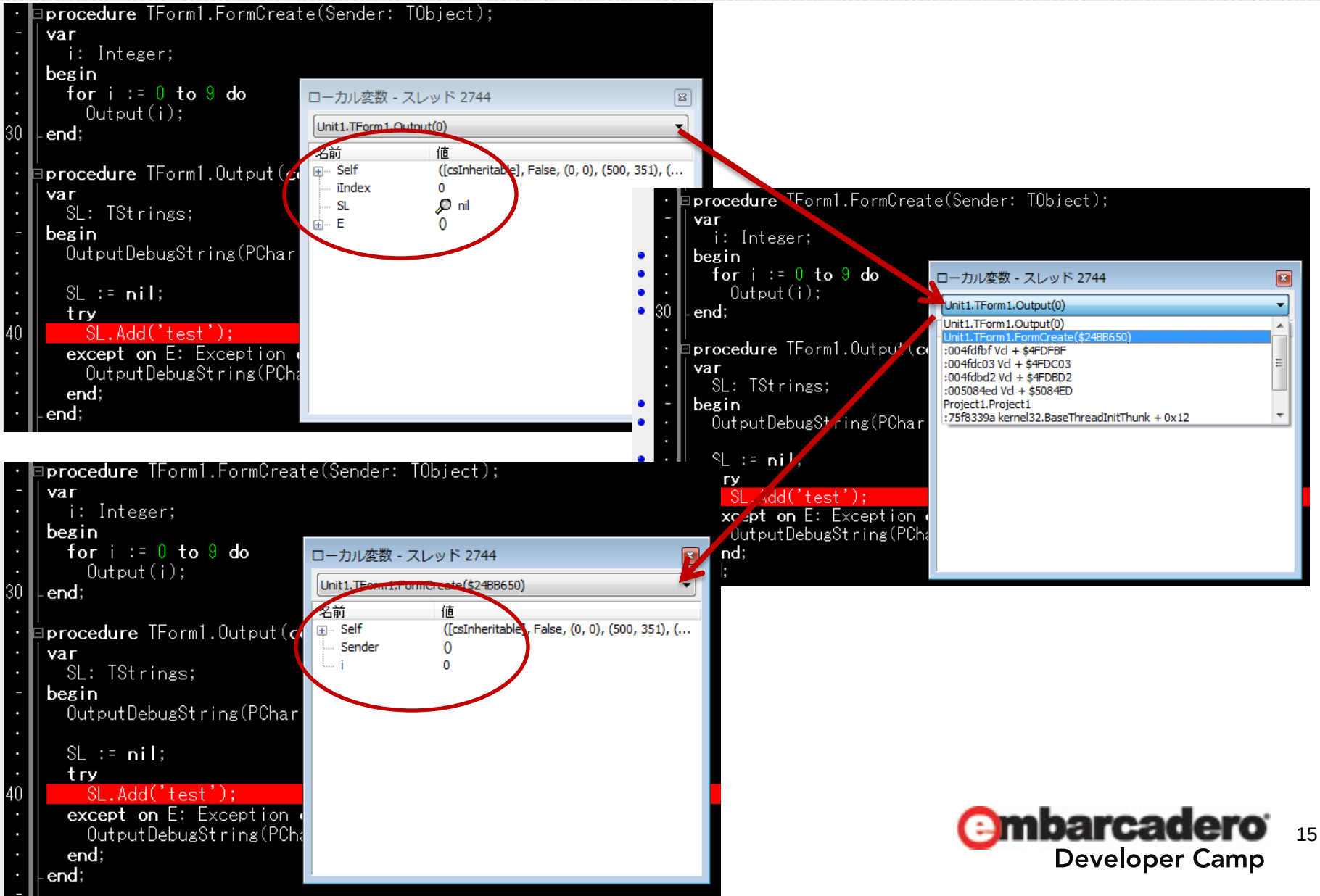
- 呼び出し履歴はいわゆる「スタックトレース」です。
- ブレークポイントで止まるまでに呼び出されたルートを記録しています



呼び出し履歴ーローカル変数

- ローカル変数ウィンドウは呼び出し履歴と密接に関連しています。
- 呼び出し履歴で呼び出された単位ごとにローカル変数を表示できます

呼び出し履歴ーローカル変数



embarcadero
Developer Camp

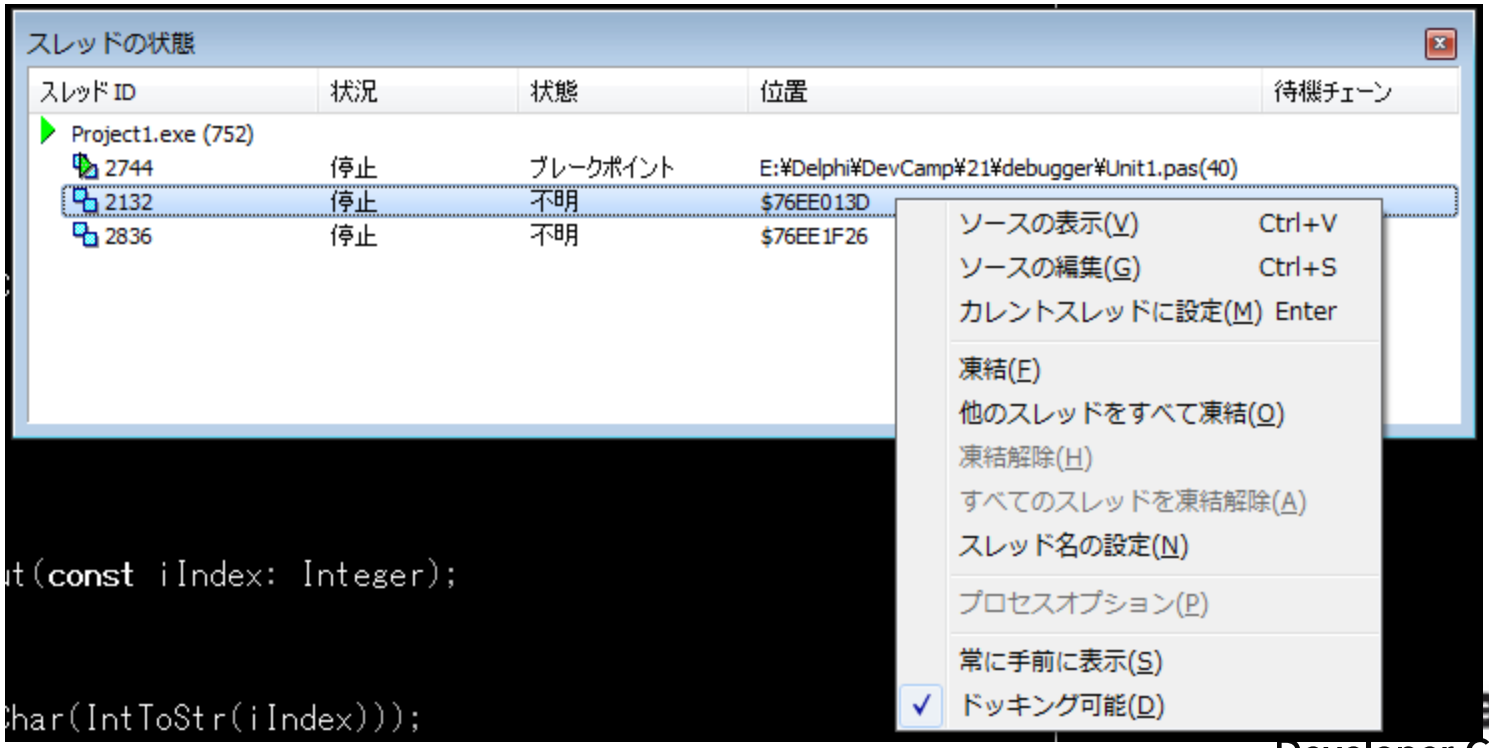
呼び出し履歴のデモンストレーション

embarcadero
Developer Camp

スレッドの状態とモジュール

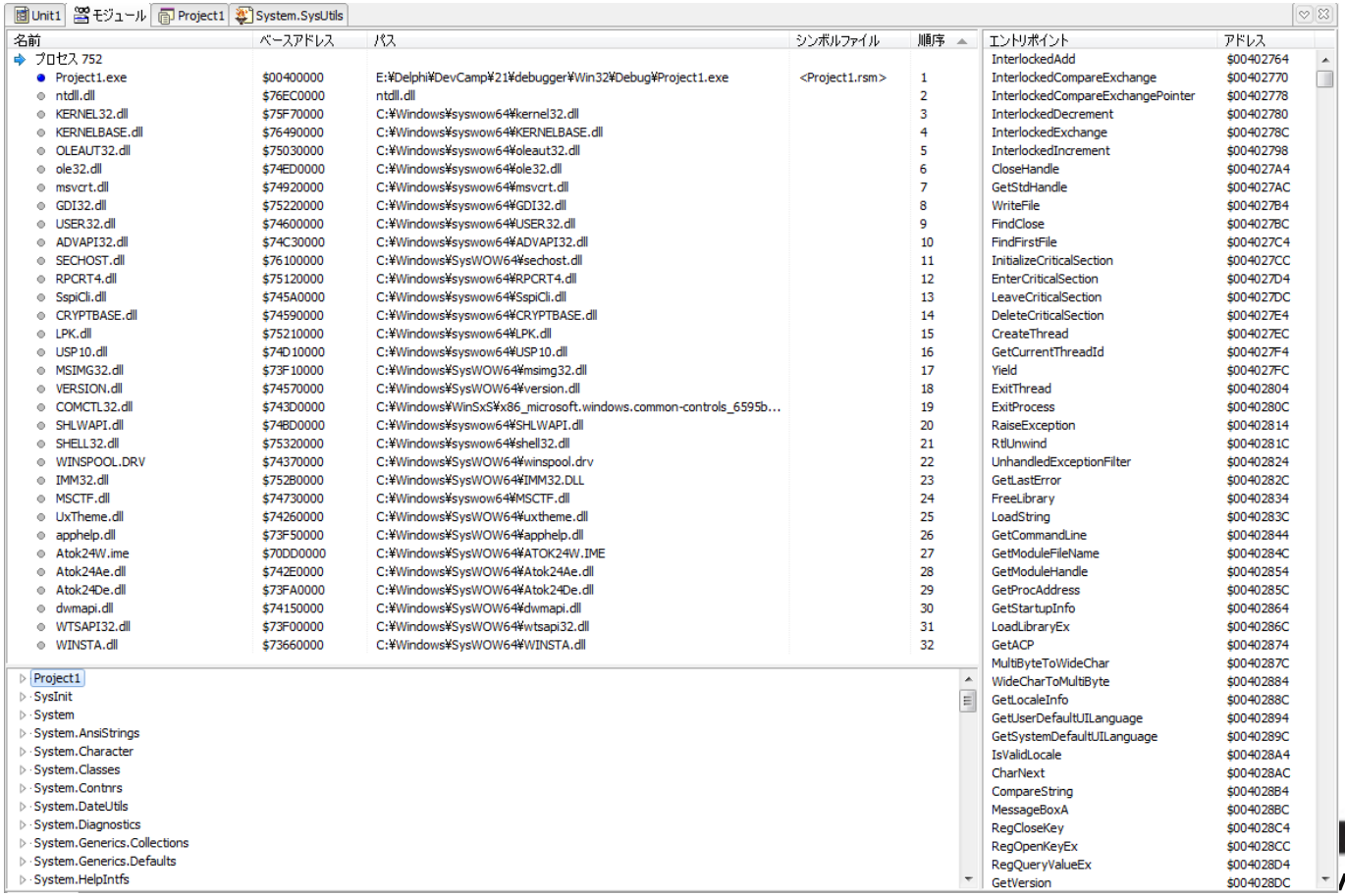
スレッドの状態

- 現在動いているスレッドの状態を示しています。
  - カレントスレッドを変更したり、実行させないように凍結したりといった操作ができます。
  - また、ブレークポイントで特定のスレッドでのみブレークさせたい場合は、このウィンドウで表示されているスレッドIDを指定できます。
    - TThread のインスタンスなど同じコードが複数のスレッドから呼ばれるときに便利です。



モジュール

- 現在のプロセスにロードされているモジュールの一覧を取得できます。
- ここで、モジュールロードブレークポイントを置くこともできます。
  - またエントリポイントの逆アセンブル表示も可能です。



embarcadero
Developer Camp

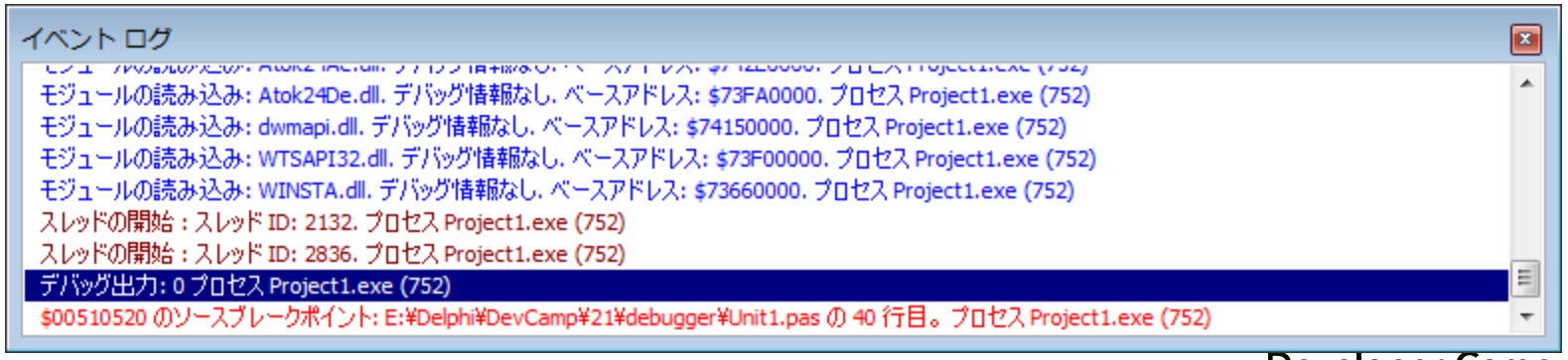
スレッドの状態とモジュールのデモンストレーション

embarcadero
Developer Camp

イベントログ

イベントログ

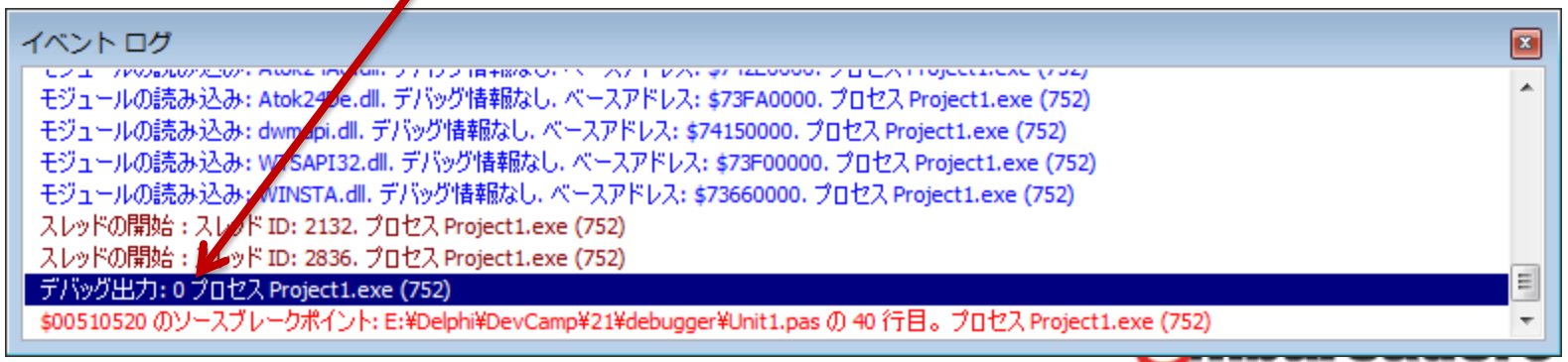
- イベントログは、気軽に使えるデバッグツールです
  - OutputDebugString を使っていわゆる「printfデバッグ」ができます
  - ブレークポイントの停止時にスタックトレースを出力することができます
  - Windowsメッセージを出力できます



イベントログ—OutputDebugString

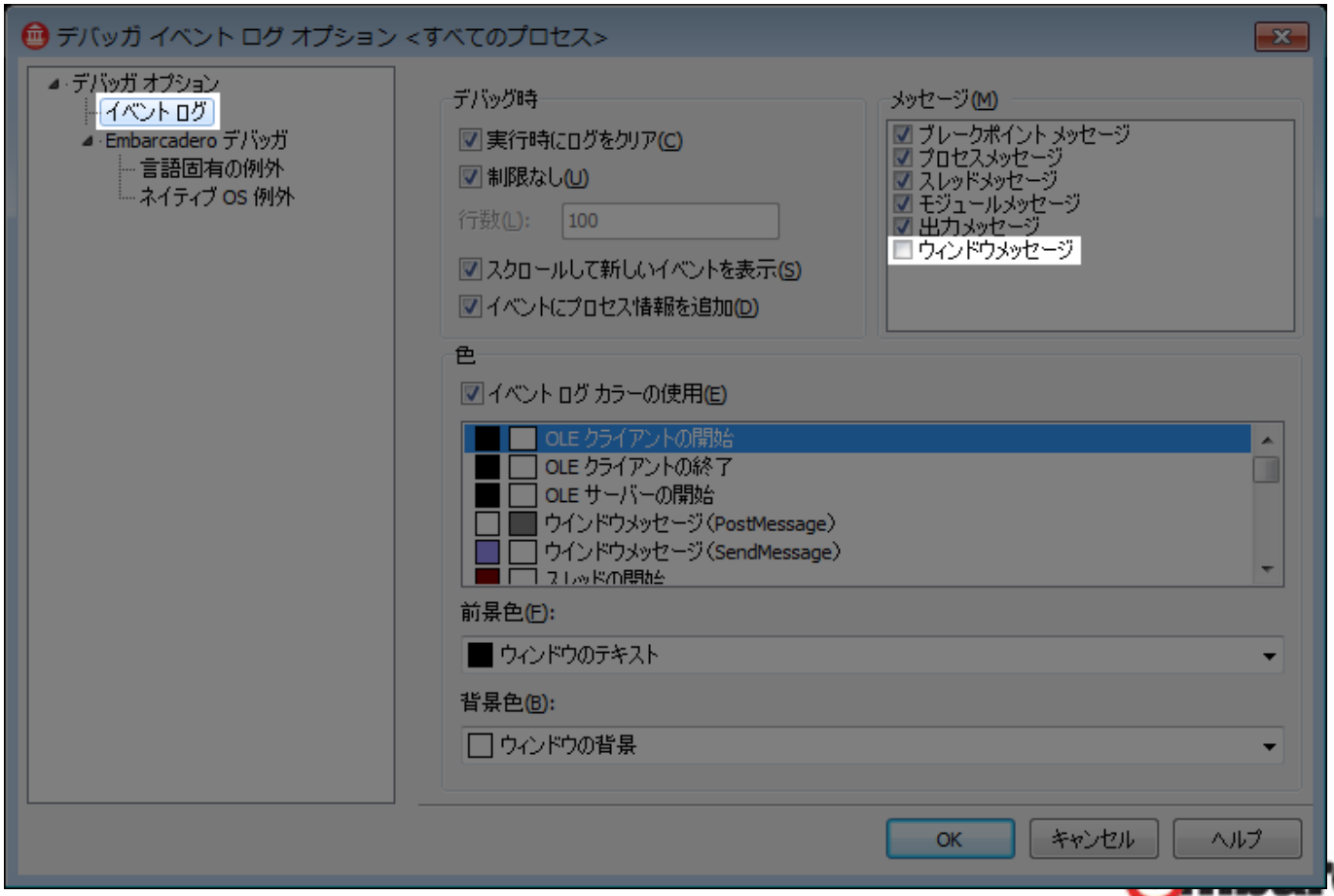
- OutputDebugString は Win32 API です。
  - 呼び出したプロセスにアタッチされているデバッガがあるときに、文字列をデバッガに渡します。
  - Delphi では当然、統合デバッガに文字列が渡ります。
    - 統合デバッガは、受け取った文字列をイベントログに出力します。

OutputDebugString を使って出力したログ



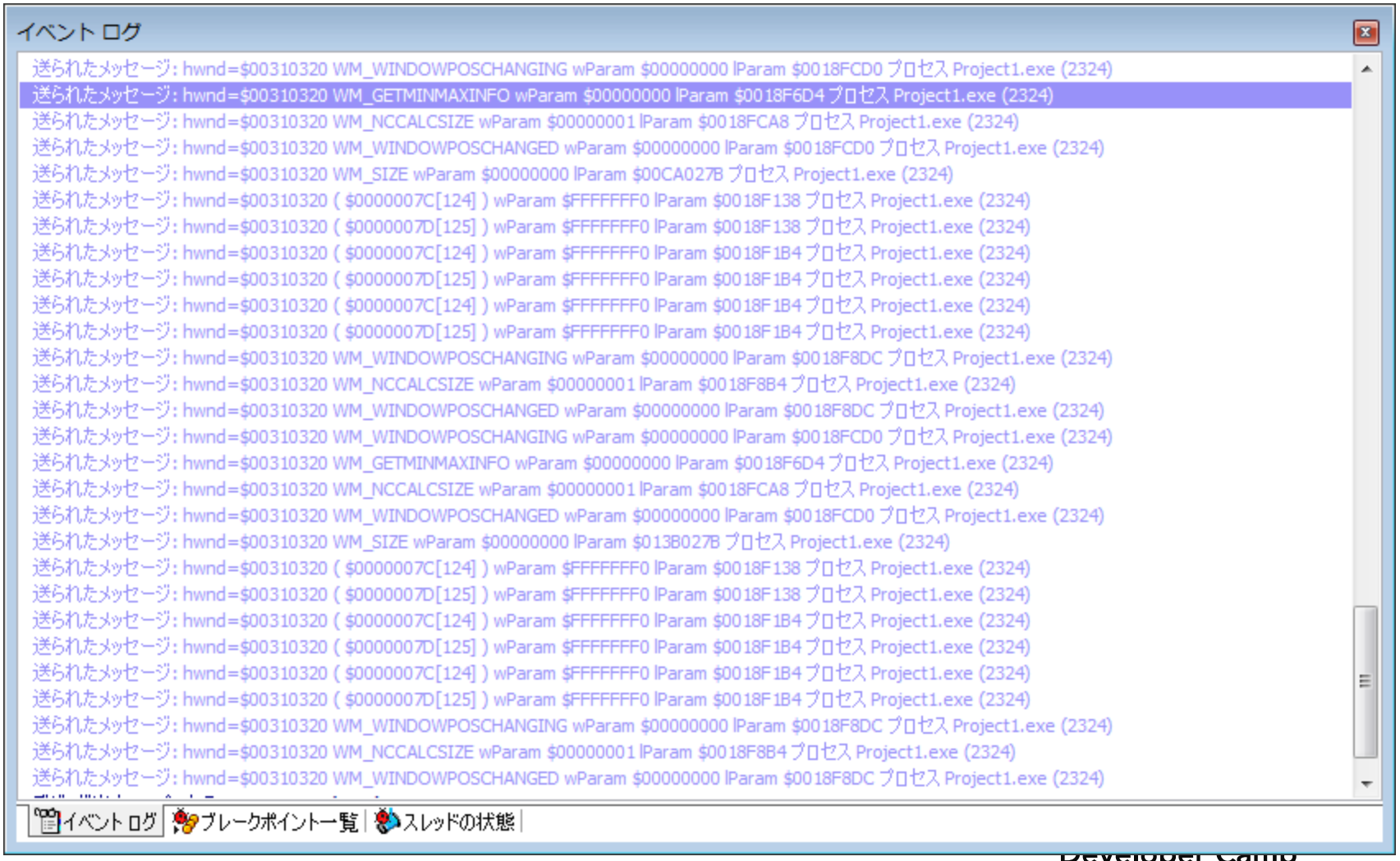
イベントログーメッセージ

- イベントログには Application が受け取ったメッセージを出力する機能があります。



イベントログーメッセージ

- 実際にメッセージを表示させると、こんな感じになります



embarcadero
Developer Camp

イベントログの
デモンストレーション

embarcadero
Developer Camp

言語で用意されるデバッグ機能

デバッグの容易さ

- モダンな言語ではデバッグ作業も重視されています。

- たとえば Java では、VM 上での動作となるため、ほぼ全ての例外をキャッチできます。キャッチした例外オブジェクトから詳細を知ることで、デバッグに生かすことができます
  - ネイティブ言語から考えると Null ポインタへのアクセスを例外としてキャッチでき、かつ、その後、続けて実行できるなんて驚異です。

- もちろん、Delphi 言語も例外なく、デバッグのための機能が用意されています。

Assert の使用

- Assert は、System ネームスペースに用意されているグローバルな手続きです。
  - 実態はコンパイラマジックによって System._Assert へ変換される特殊な手続きです。

- Assert を使うと、任意の条件で例外（EAssertionFailed）を発生させます
  - Assert の第一引数が false の時 EAssersionFailed 発生
    - 例えば Assert(Sender is TButton); とすると Sender が TButton では「無かったとき」例外を発生させます。

- $ASSERTIONS ON/OFF（もしくは $C +/-）指令を使ってコンパイラに直接 Assert のコード生成を抑制させることができます。

Assert の使用

- Assert の内部を見ると AssertionFailed 発生時に AssertErrorProc という手続きを呼んでいます。
- AssertErrorProc はグローバルな変数です。
- AssertErrorProc の中身を自分のハンドラに変えると、詳細な情報を得られます。

Assert の使用ーAssertErrorProc

```
procedure AssertErrorHandler(
  const iMsg, iFilename: String;
  const iLineNo: Integer;
  const iAddress: Pointer);
var
  Err: String;
begin
  Err :=
    Format(
      '%s (%s line %d @ %x)',
      [iMsg, iFilename, iLineNo, Integer(iAddress)]);

  ShowMessage(Err);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  AssertErrorProc = AssertErrorHandler;
  Assert(false);
end;
```

Assert の使用ートリック

- AssertErrorProc を使うとエラーが発生したユニット、行番号、アドレスなどが判ります。

- これを利用して任意の例外で、ユニットや大まかな行番号を取得できます。

Assert の使用ーAssertErrorProc

```
procedure AssertErrorHandler(
  const iMsg, iFilename: String;
  const iLineNo: Integer;
  const iAddress: Pointer);
begin
  省略
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  SL: TStringList;
begin
  AssertErrorProc = AssertErrorHandler;
  SL := nil;
  try
    SL.Add('TEST');
  except
    Assert(false); // except 内で Assert を呼び AssertErrorProc を実行させる
  end;
end;
```

embarcadero
Developer Camp

Assert の デモンストレーション

TApplication.OnException

- TApplication.OnException というイベントハンドラがあります。

- これは例外が発生したときに呼ばれ、例外処理を自前の処理に置き換えることができます。
  - デフォルトでは例外ダイアログが表示されますが、それをログファイルに出力するように書き換える事もできます。

System.NoErrMsg

- NoErrMsg について、ついでに紹介します。
- このグローバル変数を True に設定すると、エラーダイアログが出ません。
- 何らかの理由でエラーダイアログを表示したくない場合に TApplication.OnException とともに設定しておくと良いかもしれません。

その他にも

- TVirtualMethodInterceptor を使うとメソッド呼び出しの後に処理を実行できます。

- Exception.StackTrace プロパティには簡易な呼び出し履歴が入っています
  - Exception.GetExceptionStackInfoProc や Exception.GetStackInfoStringProc を置き換えたり、Exception.RaiseOuterException を利用するとサードパーティー製（自前）の例外処理を追加できます。

- class helper を使えば任意のクラスのデータを公開することもできます。

embarcadero
Developer Camp

付録：外部ツール

embarcadero
Developer Camp

EventViewer (System Log)

EventViewer

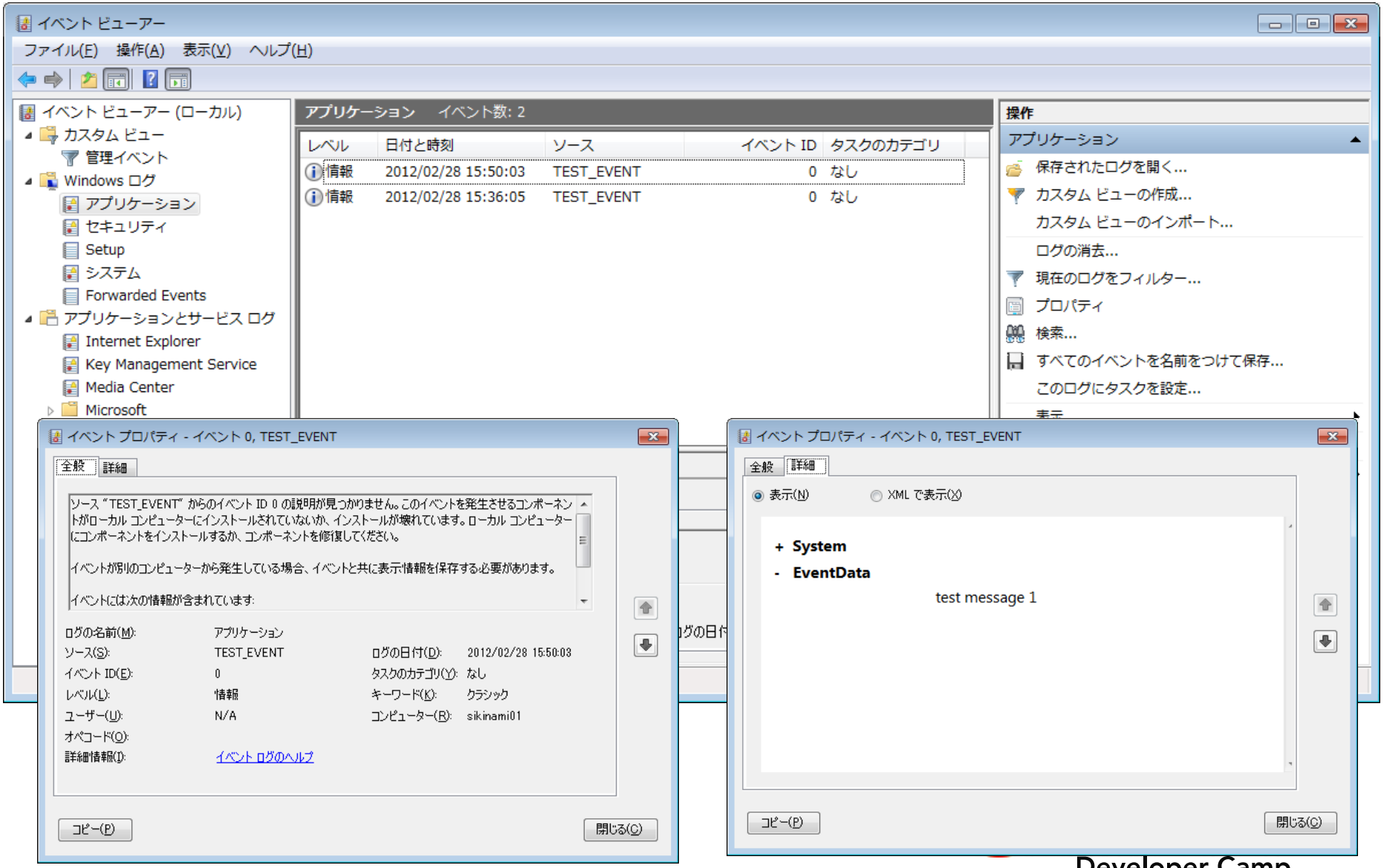
- サービスプログラムやサーバープログラムなど GUI を持たないプログラムではログが重要です
- 自前のログシステムでログを吐くことも、もちろんできますが Windows の SystemLog を使うこともできます。
- Windows の SystemLog はイベントビューアで見ることができます。

EventViewer一書き込む（ソース抜粋）

```
procedure TEventLog.Add (
  const iLogType: TLogType;
  const iCategory, iEventId: Integer;
  const iMsg: String;
  const iDataLength: Integer;
  const iData: Pointer);
var
  Ptr: Pointer;
begin
  Ptr := PChar(iMsg);

  FHandle := RegisterEventSource(nil, PChar(FName));
  try
    ReportEvent(
      FHandle,
      Ord(iLogType), // Event Type
      0,             // Category
      0,             // Event Id
      nil,           // User Sid
      1,             // 書き込む文字列の数
      iDataLength,   // 書き込むバイナリデータのバイト数
      @Ptr,          // 書き込む文字列（ポインタのポインタ）
      iData);        // バイナリデータ
  finally
    DeregisterEventSource(FHandle);
  end;
end;
```

EventViewer―書き込んだ情報の閲覧



Developer Camp

embarcadero
Developer Camp

EventViewer のデモンストレーション

embarcadero
Developer Camp

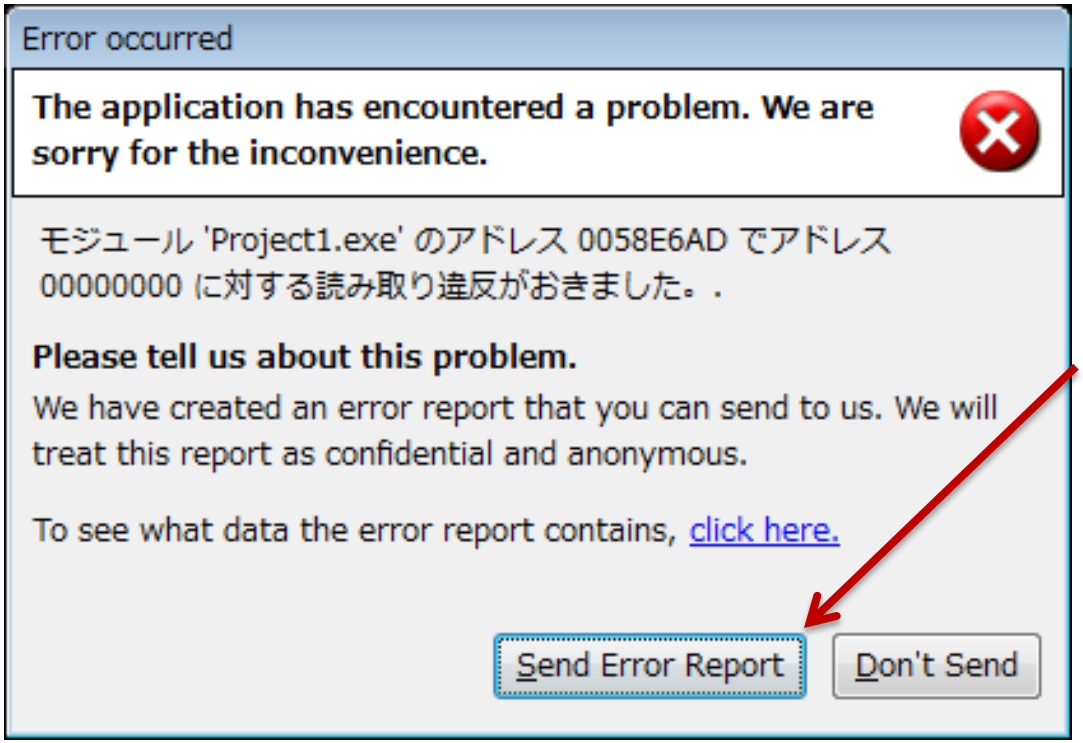
EurekaLog

EurekaLog

- EurekaLog はパワフルなデバッグ用ツールです
- 全ての例外、無限ループ、メモリの未解放などを検出できます
- しかも！検出した例外はスタックトレース、スクリーンショット、逆アセンブル結果などの情報付きで実行先から送信可能です。
- 入手先：http://eurekalog.com/
  - シングルデベロッパーライセンスで 99ドル

EurekaLog

- エラーが発生するとエラーダイアログが表示されますがEurekaLog を組み込んだ実行ファイルでは EurekaLog のエラーダイアログが表示されます

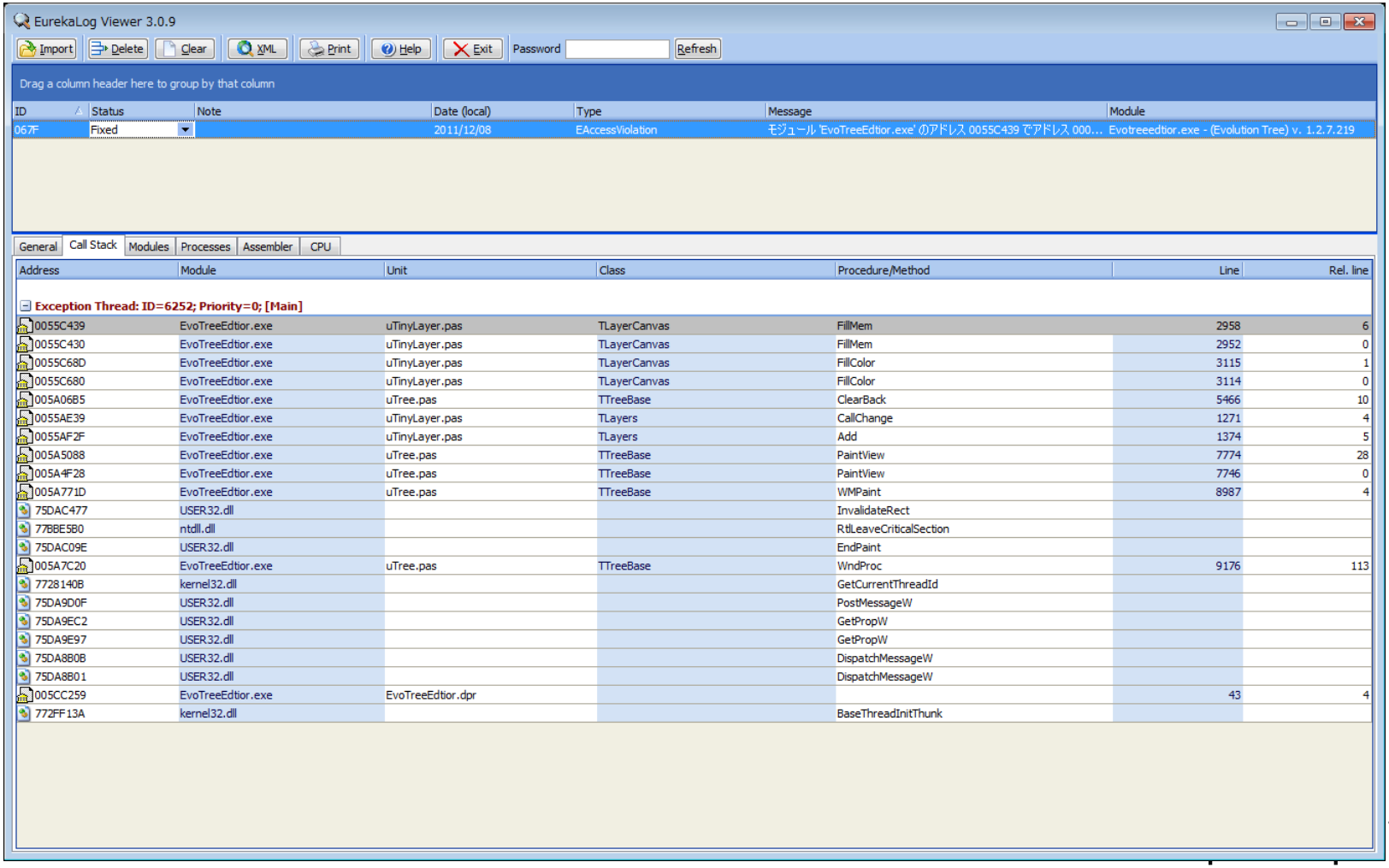


ちなみにちゃんと設定すれば、このダイアログに出る文言はローカライズできます

email クライアントで添付として送ったり、EurekaLog が搭載する SMTP サーバを介して送ったり、BugZilla 形式のウェブにポストしたりと色々な形式でバグリポートを送信できます

EurekaLog

- EurekaLog が送ってきたレポートは、専用のビューアでこんな感じに表示されます



embarcadero
Developer Camp

EurekaLog の デモンストレーション

embarcadero
Developer Camp

Q & A