embarcadero
Developer Camp

【T8】テクニカルセッション

「Delphi言語再入門」～拡張されたRTTIを試してみる

東洋テクニカルシステム株式会社
システム開発部
福士　光

embarcadero
Developer Camp

アジェンダ

アジェンダ

- 従来のRTTIでできること
- Delphi 2010で新しく拡張されたRTTI(拡張RTTI)でできるようになったこと
- 試してみる(1)～クラス内のメンバの列挙
- 試してみる(2)～クラス内のメンバの値の取得
- 拡張RTTIと属性を使う上での注意事項

embarcadero
Developer Camp

基本知識

基本知識～RTTIとは？

- RTTI(実行時型情報)とは？
  - RunTime Type Informationの略(Informationではなく Identificationとすることもある)。
  - 実行時にメモリ上のオブジェクトのデータ型に関する 情報を取得、操作できる。
  - Delphiでは従来からpublished宣言を行うことでRTTIが生成され、IDEがコンポーネントの持つRTTIを利用している。
  - Delphi 2010/XE/XE2ではRTTIが拡張され、より多くのことが可能になった(実行バイナリのサイズが大きくなるという副作用がある)。

embarcadero
Developer Camp

従来のRTTIでできること

従来のRTTIでできること(1)

- (System.)TypInfoユニット
- 実行時にプロパティの型情報を探す。
  - publishedなプロパティのみ(コンパイラ指令{$M+}または {$TYPEINFO ON}が必要)
  - GetPropInfo関数、GetPropList関数など
  - PPropInfo = ^TPropInfo (レコード型へのポインタ)
- 実行時に型情報を元に値を取得、設定する。
  - GetXXXXProp/SetXXXXProp関数(XXXXは対象となるプロパティの型による)
- DelphiのIDEがフォームをストリーム化したりプロパティエディタで使用しているのはご存知のとおり。

従来のRTTIでできること(2)

- 仮想メソッドテーブル(VMT)
  - virtual/overrideと宣言されたメソッドのテーブル(クラスごとに存在)
- 動的なメソッドのテーブル
  - dynamic/overrideあるいはmessageと宣言されたメソッドのテーブル(テーブル上では動的メソッドなのかメッセージハンドラなのかは区別されていない)

embarcadero
Developer Camp

拡張RTTIでできるようになったこと

拡張RTTIでできるようになったこと(1)

- (System.)RTTIユニット
- 実行時にフィールド、プロパティ、メソッドの型情報を取得する。
  - コンパイラ指令{$RTTI}で拡張RTTIをプロパティ、フィールド、メソッドのそれぞれに対してどの範囲(published/public/protected/private)のものに付けるのかを制御
  - クラス型(class)またはレコード型(record)が対象

    →型情報そのものは全ての型(Integer, Boolean, ...)に存在している

拡張RTTIでできるようになったこと(2)

- 実行時に型情報とインスタンスへのポインタを元に各種の操作を行う。
  - フィールドの値の取得、設定
  - プロパティの値の取得、設定
  - メソッドの呼び出し(invocation)

拡張RTTIでできるようになったこと(3)

- 属性(attribute)による注釈付け(annotation)
  - クラス型あるいはレコード型に属性で注釈を付ける
  - クラス型あるいはレコード型のメンバ(フィールド、プロパティ、メソッド)に属性で注釈を付ける
    - .NET Frameworkと同様の記法を使う

```
[<Attr>]
[<Attr>(<parameterlist>)]
```

    - 先頭の“T”と末尾の“Attribute”、コンストラクタの“.Create”を省略できる

```
[T<Attr>Attribute.Create]
[T<Attr>Attribute.Create(<parameterlist>)]
```

拡張RTTIでできるようになったこと(4)

- 属性(attribute)による注釈付け(annotation)
  - カスタム属性(custom attribute)の宣言
    - (プロパティやフィールドの値ではなく)属性クラスの型で区別する

      →例外ハンドラを記述するときに例外オブジェクトの型で区別を行うのと同様に
    - コンストラクタで渡した値(定数のみ)をフィールドまたはプロパティに保存して参照することもできる
    - `TCustomAttribute`クラスから派生したカスタム属性クラスを宣言して使用する

拡張RTTIでできるようになったこと(5)

- 属性(attribute)による注釈付け(annotation)
  - 実行時にクラス型、レコード型に付けられている属性を抽出する
  - 実行時にクラス型、レコード型のメンバに付けられている属性を抽出する

- 属性を使う状況
  - 同じ型から派生していても区別して処理したい
  - 同じ型のメンバでも区別して処理したい

拡張RTTIでできるようになったこと(6)

- 拡張RTTIと属性についての補足。
  - 拡張RTTIはデフォルトでは以下の範囲に付けられている(Systemユニットで定義)

可視性	private	protected	public	published
フィールド	○	○	○	○
プロパティ	×	×	○	○
メソッド	×	×	○	○

  - Delphi 2010以降の実行ファイルが大きくなってしまう原因のひとつ
  - {$RTTI EXPLICIT ...}で(継承元クラスの指定とは独立して)拡張RTTIを付ける範囲を指定できる

拡張RTTIでできるようになったこと(7)

- 拡張RTTIと属性についての補足。
  - 属性は検索、抽出されるときに実体が生成される
    →検索、抽出しなければ性能上のペナルティはない
    →実行バイナリ、占有メモリのサイズのペナルティはある
  - 拡張RTTIを扱うコードは遅い

- どのような場合に拡張RTTIを使えばよいのか？
  - クラスに対する汎用な処理の記述
    →ORマッパやXMLへのシリアライズ/デシリアライズ
    →クラス構造のツリー表示

embarcadero
Developer Camp

試してみる(1)
クラス内のメンバの列挙

クラス内のメンバの列挙(1)

- TRTTIContext ((System.)RTTIユニット)
  - 全ての操作はここから始まる
  - 高度なレコード型
  - 内部リソースの管理、解放のためクラスのコンストラクタ、デストラクタのようにclass function Createとprocedure Freeを呼ぶことが推奨されています

```
uses
  Rtti;

var
  ctx: TRttiContext;
begin
  ctx := TRttiContext.Create;
  try
    // RTTIを扱う
  finally
    ctx.Free;
  end;
end;
```

クラス内のメンバの列挙(2)

- TRTTIContext
  - GetTypeメソッド
    - 指定されたクラス型のRTTIオブジェクト(TRTTITypeから派生したクラスのインスタンス)を取得

```
function GetType(AClass: TClass): TRttiType;
```

クラス内のメンバの列挙(3)

- TRTTIType
  - クラス型(RTTIオブジェクトの基底クラス)
  - GetPropertiesメソッド
    - 所属するクラスのプロパティのRTTI情報を全て取得

```
function GetProperties: TArray<TRttiProperty>;
```

  - GetFieldsメソッド
    - 所属するクラスのフィールドのRTTI情報を全て取得

```
function GetFields: TArray<TRttiField>;
```

  - GetMethodsメソッド
    - 所属するクラスのメソッドのRTTI情報を全て取得

```
function GetMethods: TArray<TRttiMethod>;
```

  - 階層順に(継承先から継承元に向かって)RTTI情報がリストアップされる

クラス内のメンバの列挙(4)

- TRTTIType
  - GetDeclaredPropertiesメソッド
  - GetDeclaredFieldsメソッド
  - GetDeclaredMethodsメソッド
  - そのクラスで定義したプロパティ/フィールド/メソッドのRTTI情報だけがリストアップされる

クラス内のメンバの列挙(5)

- TRTTIProperty
  - クラスのプロパティのRTTI情報
  - クラス型(TRTTIMemberから派生)
  - Nameプロパティ(名前)

```
property Name: string;
```

  - Visibilityプロパティ(可視性)

```
property Visibility: TMemberVisibility;
```

クラス内のメンバの列挙(6)

- それでは実際に試してみましょう。
  - 新規作成→VCLフォームアプリケーション
  - フォーム上にボタン(Button1)とメモ(Memo1)を配置
  - メモのAnchorsにakRightとakBottomを追加、crollBarsをssBothに変更。

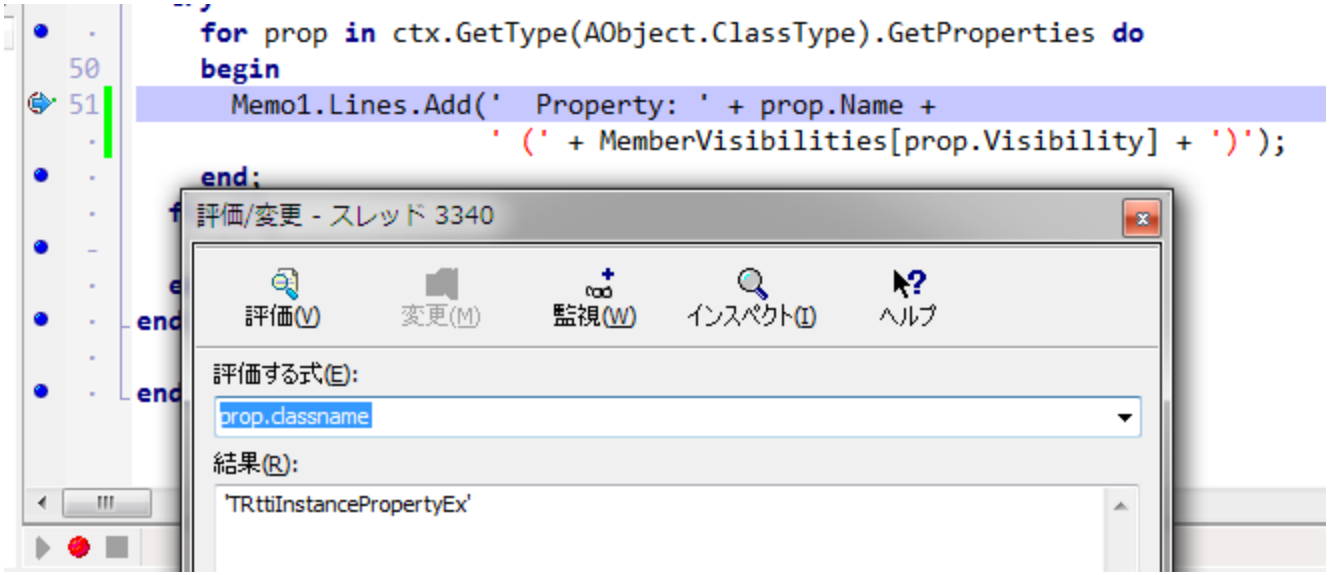
クラス内のメンバの列挙(7)

```
procedure TForm1.EnumProperties(AObject: TObject);
const
  MemberVisibilities: array [TMemberVisibility] of String =
    ('private', 'protected', 'public', 'published');
var
  ctx: TRttiContext;
  prop: TRttiProperty;
begin
  Memo1.Lines.Add('Class: ' + AObject.ClassName);
  ctx := TRttiContext.Create;
  try
    for prop in ctx.GetType(AObject.ClassType).GetProperties do
    begin
      Memo1.Lines.Add('  Property: ' + prop.Name +
                      ' (' + MemberVisibilities[prop.Visibility] + ')');
    end;
  finally
    ctx.Free;
  end;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  EnumProperties(Form1);
  EnumProperties(Button1);
  EnumProperties(Memo1);
end;
```

クラス内のメンバの列挙(8)

- とりあえず実行してみましょう。
- Memo1.Lines.Addの行にブレークポイントを設定し、prop(TRttiProperty)を評価してみる。
- propの実際の型は？



embarcadero
Developer Camp

試してみる(2)
クラス内のメンバの値の取得

クラス内のメンバの値の取得(1)

- TValue ((System.)RTTIユニット)
  - 拡張RTTIでデータを格納する
  - 高度なレコード型
  - バリアントもどき(“バリアント型の軽量版”)
  - 実際のデータはFDataフィールド(TValueDataレコード型、共用体)に格納している

クラス内のメンバの値の取得(2)

- TValue
  - Kindプロパティ
    - 型の種類を列挙として取得

```
property Kind: TTypeKind;
```

  - TypeInfoプロパティ
  - TypeDataプロパティ
    - 型の情報をレコード型として取得

```
property TypeInfo: PTypeInfo read GetTypeInfo;
property TypeData: PTypeData read GetTypeDataProp;
```

クラス内のメンバの値の取得(3)

- TValue
  - IsXXXXメソッド/プロパティ
    - 格納されているデータの状態を問い合わせる

    IsEmpty/IsObject/IsInstanceOf/IsClass/IsOrdinal/IsType/IsArray
  - AsXXXX/TryAsXXXXメソッド
    - 格納されているデータを特定の型で取得する

    AsObject/AsClass/AsOrdinal/AsType/AsInteger/AsBoolean
    AsExtended/AsInt64/AsInterface/AsString/AsVariant/AsCurrency
    TryAsOrdinal/TryAsType

クラス内のメンバの値の取得(4)

- TValue
  - 暗黙の型変換(implicit conversion)
    - データを格納する

    string/Integer/Extended/Int64/TObject/TClass/Boolean
  - FromXXXXメソッド
    - データを格納する

    FromVariant/From<T>/FromOrdinal/FromArray
  - ToStringメソッド
    - データをとりあえず文字列化

クラス内のメンバの値の取得(5)

- TRTTIPropertyとTValue
  - TRTTIProperty.GetValue
    - 特定のインスタンスのプロパティの値を取得

```
function GetValue(Instance: Pointer): TValue;
```

  - TRTTIProperty.SetValue
    - 特定のインスタンスのプロパティの値を設定

```
procedure SetValue(Instance: Pointer; const AValue: TValue);
```

  - TRTTIPropertyが示しているのは型に関する情報であることに注意

クラス内のメンバの値の取得(6)

- こちらも試してみましょう。
  - GetValueでプロパティの値を取得して、TValue.ToStringで文字列化します
  - GetValueは例外が起きるかもしれないのでtry...except...endで保護します

```
procedure TForm1.EnumProperties(AObject: TObject);
const
  MemberVisibilities: array [TMemberVisibility] of String =
    ('private', 'protected', 'public', 'published');
var
  ctx: TRttiContext;
  prop: TRttiProperty;
  V: TValue;
  Str: String;
```

クラス内のメンバの値の取得(7)

```
begin
  Memo1.Lines.Add('Class: ' + AObject.ClassName);
  ctx := TRttiContext.Create;
  try
    for prop in ctx.GetType(AObject.ClassType).GetProperties do
    begin
      try
        V := prop.GetValue(AObject);
        Str := V.ToString;
      except
        on E: Exception do
        begin
          Str := 'Error (' + E.Message + ')';
        end;
      end;
      Memo1.Lines.Add('  Property: ' + prop.Name +
                      ' (' + MemberVisibilities[prop.Visibility] + ')' + ' Value = ' + Str);
    end;
  finally
    ctx.Free;
  end;
end;
```

クラス内のメンバの値の取得(8)

- SetValueメソッドでプロパティの値を変更することも可能です。
- GetValueで取得したTValueのTypeInfo.KindがtkClassなら、AsObjectはクラス型のプロパティです。
- あるいはTRTTIPropertyやTRTTIFieldのHandleプロパティがnilでなければPTypeInfo(=^TTypeInfo)なので、Handle.Kindでそのプロパティ/フィールドの定義の型を知ることもできます。

  →再帰処理で複合クラスのトラバースが実現できます。

embarcadero
Developer Camp

拡張RTTIと属性を使う上での注意事項

拡張RTTIと属性を使う上での注意事項(1)

- 配列に対するサポートが弱い。
  - Delphi XE2ではTRttiTypeにGetIndexedPropertiesメソッドが追加されて配列プロパティに関する情報を取得できるようになったが、通常のプロパティに比べて微妙に使えない
  - 静的配列のフィールドもうまく扱えない
  - 動的配列のフィールドは通常のプロパティ並に扱えるので、配列プロパティ、静的配列のフィールドの代替として動的配列のフィールドを用意してエイリアス的に使うという回避策が有効

拡張RTTIと属性を使う上での注意事項(2)

- 属性のコンストラクタ
  - TCustomAttributeのコンストラクタはパラメータを持たないが、派生したクラスでコンストラクタを定義することで値を渡すことができる(フィールド、プロパティでその値を保持する)

```
constructor Create(const AFooValue: String);
```

  - しかしコンストラクタのパラメータは定数しか使えない(文字列定数はOK)
  - ポインタでも定数なら使えるはずだが、現実には内部エラーが発生してコンパイルできない

embarcadero
Developer Camp

参考文献

参考文献 (1)

- **Delphiのヘルプ**

  RTTI の操作（オンライン）

  http://docwiki.embarcadero.com/RADStudio/ja/RTTI_%E3%81%AE%E6%93%8D%E4%BD%9C%EF%BC%9A%E3%82%A4%E3%83%B3%E3%83%87%E3%83%83%E3%82%AF%E3%82%B9

- **“Rob’s Technology Corner”**

  “Delphi 2010 - RTTI & Attributes”シリーズ

  http://robstechcorner.blogspot.com/2009/09/so-what-is-rtti-rtti-is-acronym-for-run.html

  – 英語です。

参考文献 (2)

- **Delphi 2010 Handbook**
  - Marco Cantù著
  - CreateSpace (http://www.createspace.com/)
  - ISBN978-1450597265 (ISBN1450597262)
  - 43.50USD(書籍版)、28.00USD(ebook)

    http://www.marcocantu.com/dh2010/

    http://www.amazon.com/dp/1450597262 (書籍)

    http://sites.fastspring.com/wintechitalia/product/delphi2010handbook (ebook/PDF)
  - 英語ですが表現は比較的平易。
  - 拡張RTTIについて記述があります(Chapter 3、4)。

参考文献 (3)

- **Delphiクイックリファレンス**
  - Ray Lischner著
  - 光田 秀、竹田 知生訳
  - オライリー・ジャパン
  - ISBN978-4873110400 (ISBN 4-87311-040-8)
  - 4,725円

    http://www.oreilly.co.jp/books/4873110408/

    http://www.amazon.co.jp/dp/4873110408
  - 残念ながら絶版です。
  - 従来のRTTIについて記述があります。

参考文献 (4)

- **Inside Delphi**
  - Ray Lischner著
  - 光田 秀訳
  - 大野 元久、服部 誠監修
  - アスキー
  - ISBN978-4756119513 (ISBN 4-7561-1951-4)
  - 9,240円

    `http://www.amazon.co.jp/dp/4756119514`
  - こちらも残念ながら絶版です。
  - 従来のRTTIについて記述があります。

embarcadero
Developer Camp

Q & A

embarcadero
Developer Camp

ありがとうございました