



**DelphiDay**  
italian conference

**Delphi Attributes, RTTI  
and real life usage cases**



# Dmitry Arefiev

Embarcadero Technologies -  
Programmer

@ darefiev@gmail.com



11-12 Giugno 2024  
Piacenza





# AGENDA

---

1. Overview
2. Declaring Attributes
3. Specifying Attributes
4. Reading Attributes
5. Examples



# Overview

1





# Overview / Delphi Help

---

- Attributes are a language feature in Delphi that allows annotating types and type members with special objects that carry additional information.
- This information can be queried at run time.
- Attributes extend the normal Object-Oriented model with Aspect-Oriented elements.



# Overview / Applications

---

Common applications are:

- Describing. For example, a human readable description is associated with a class property.
- Binding. For example, a record field is associated with a DB field name.
- Processing rules. For example, a class property is marked as optional.



# Overview / Example

```
TCustomer = record  
  [DBField('fullname'), Readable('The name', 'The customer full name')]  
  FName: string;  
end;
```

- FName field is associated with “FullName” DB field and is described.
- Without attributes: you can do the same in code, using a dictionary.
- With attributes: you are concentrated on the data structure, methods, etc.



# Declaring Attributes

# 2





# Declaring / TCustomAttribute

- An attribute is a class, inherited from **TCustomAttribute**
  - ◆ If not, then **E2010** error.
- An attribute class name should be **<name>Attribute**
  - ◆ If yes, then use [name], eg [Readable]
  - ◆ if not, then use full class name, eg [DBFieldAttr]

```
DBFieldAttr = class(TCustomAttribute);  
ReadableAttribute = class(TCustomAttribute);  
TCustomer = record  
    [DBFieldAttr, Readable]  
    FName: string;  
end;
```



# Declaring / Constructors

- Usually an attribute class has constructors.
- RTL calls constructor to create attribute object.
- Actually, attribute specification is like a constructor call without **Attribute.Create** part
- Arguments must be of ordinary types, otherwise **E2026** error

```
DBFieldAttribute = class(TCustomAttribute)
public
    constructor Create(const AName: string); overload; // OK
    constructor Create(const AField: TField); overload; // Bad
end;
```



# Specifying Attributes

# 3



# Specifying / Syntax

---

Attributes are listed before the annotatable item, using one of the following syntaxes (or mixed):

```
TCustomer = record
  [DBField('fullname'), Readable('The name', 'The customer full name')]
  FName: string;
end;
```

```
TCustomer = record
  [DBField('fullname')]
  [Readable('The name', 'The customer full name')]
  FValue: string;
end;
```



# Specifying / Notes

---

- Attribute specification is a constructor call without **Attribute.Create** part.
- An argument value must be a simple expression, evaluated at compile time.
- Unknown attribute leads to W1074 warning, not error. And the attribute is excluded from RTTI.
- C++ Builder does not support Delphi attribute specifying.



# Reading Attributes

4





# Reading / System.Rtti

---

- Attributes are embedded into RTTI and saved into .dcu
- At run time they may be fetched using **System.Rtti**
- All RTTI classes (**TRttiType**, **TRttiField**, **TRttiProperty**, etc) have methods (introduced in **TRttiObject**):

```
function GetAttribute(AAttrClass: TCustomAttributeClass): TCustomAttribute;  
function GetAttribute<T: TCustomAttribute>: T;  
function HasAttribute(AAttrClass: TCustomAttributeClass): Boolean;  
function HasAttribute<T: TCustomAttribute>: Boolean;  
function GetAttributes: TArray<TCustomAttribute>;
```



# Reading / Notes

---

```
var
  LCtx: TRttiContext;
  LType: TRttiType;
  LAttr: DBFieldAttribute;

LCtx := TRttiContext.Create;
LType := LCtx.GetType(TypeInfo(TCustomer));
LAttr := LType.GetAttribute<DBFieldAttribute>;
```

- Do not not release the attribute objects. This is applicable to all RTTI objects, eg **TRttiType**.
- You should be careful, when an application is compiled with run-time packages, and loading/unloading them.



# Reading / Hints

---

- Reading is a relatively expensive operation. When performance is critical, consider to cache attribute infos.
- To associate a Boolean value, use an “empty” attribute class, specified only when value is True.

```
OptionalAttribute = class(TCustomAttribute);  
TMyClass = class  
    [Optional]  
    FValue: Integer;  
end;  
...  
LField.HasAttribute<OptionalAttribute>;
```



Examples.

5



# Annotating Properties in Inspector

- Implements user friendly **PropInspect.TfrmPropInspect** form
- Uses JEDI VCL Property Inspector component  
(<https://github.com/project-jedi/jvcl>)
- Introduces **ReadableAttribute** associating human readable name and description with an object property.
- Uses PI **AfterItemCreate** to provide a property readable name.
- Uses PI **OnItemSelected** to show a property readable description.



# Persisting Object in INI file

---

- Implements **INIReaderWriter.TObjectIniReaderWriter** class
- Introduces **INIAttribute** allowing to associate INI file section and value names with an object properties.
- Only properties with **[INI(...)]** attribute are read/written.
- Uses existing RTL **DefaultAttribute** to assign a default value to a property when INI file has no value for this property.
- Performance is low critical.





# Persisting Records in Database

---

- Implements **DataSetReader.TRecordDataSetReader<T>** class
- Introduces **DBFieldAttribute** associating a dataset and record fields.
- All record fields are loaded from dataset, if not disabled for a record field by **[DBField]** attribute.
- To improve performance **TRecordDataSetReaderBase.Bind** method is called only once for a dataset and record type.



# Industrial Automation Application

---

- This is a proprietary industrial automation framework for Delphi developed by Technolog SRL company.
- It is controlling industrial hardware, eg cranes, conveyors, AGV's, which are connected through PLC controllers to PC.
- Framework lets to bind a Delphi class to a PLC controller using **[PLC(...)]**, etc attributes.
- In this kind of applications the performance is very critical.



THANK YOU