DelphiDay italian conference PIACENZA EDITION

2019 - XVIII Edizione

OPENAPI 3.0

Document your REST API

wintech italia

embarcadero reseller partner

Paolo Rossi

Wintech Italia **CTO**

Delphi **Dev**

Web Dev

DelphiDay italian conference

PIACENZA EDITION

2019 - XVIII Edizione

wintech italia

AGENDA

- → Documenting REST API, why?
- → Delphi REST libraries with documentation
  - ◆ WiRL, RAD Server
- → Swagger (OpenAPI 2)
- → OpenAPI 3.0
- → OpenAPI-Delphi
  - ◆ Classes
  - ◆ Serialization
    - ● Neon Library

GITHUB PROJECT

OPENAPI 3.0 DELPHI LIBRARY

Scan me

DelphiDay italian conference

PIACENZA EDITION

2019 - XVIII Edizione

wintech italia

GITHUB PROJECTS

WiRL Project

JAX-RS Like REST Library for Delphi

OPENAPI 3.0 DELPHI LIBRARY

OpenAPI-Delphi

OpenAPI 3.0 library for Delphi

DelphiDay italian conference

PIACENZA EDITION

2019 - XVIII Edizione

wintech italia

GITHUB PROJECTS

Delphi JWT

JSON Web Token Library for REST
(and not only REST)

Delphi Neon

JSON Serialization Library for REST
(and not only REST)

DelphiDay italian conference
PIACENZA EDITION

2019 - XVIII Edizione

wintech italia

REST API

REST API

- Q1: have you made a REST service?
  - Q1.2: what REST library do you use?
  - Q1.1: if not, are you considering making one?
- Q2: for what purpose?

REST API

- REST services can be consumed by other
  - Not in your company
- You have to document these APIs in order to be used by others
  - Maybe not if you implement HATEOS
- How to document a REST API?

REST API

- You can write and keep in sync the documentation by yourself
  - But what if the REST service is able to document itself?
- What are the tools to document my API?

SWAGGER TOOLS

SWAGGER TOOLS

- ➔ SmartBear company
- ➔ Swagger is a tool (a set of tools)
- ➔ Specification for the documentation have been renamed to OpenAPI (more later)
- ➔ Swagger Editor
- ➔ Swagger UI

SWAGGER EDITOR

- → API editor for designing APIs with the OpenAPI Specification
- → Can be run locally or accessed on the Web

https://editor.swagger.io/

SWAGGER UI

- ➔ Visualize OpenAPI specification in an interactive UI
- ➔ Collection of HTML, JS, and CSS that dynamically generate documentation from a Swagger-compliant API

https://github.com/swagger-api/swagger-ui

RAD SERVER

RAD SERVER

- Demo in:
  - Object Pascal\Database\EMS\APIDocAttributes
- Based on resource’s attributes
- Based on OpenAPI 2.0 (ex swagger)
- Waiting for OpenAPI 3.0 support

WIRL

WIRL

- WiRL extracts information from the resource and resource's method
- Based (currently) on OpenAPI 2.0
- Plan to support OpenAPI 3.0 (very soon)
- Additional info (will be) based on XMLDoc as well as custom attributes
  - I don't like very much polluting the code with attributes that aren't code-related

OPENAPI 3.0

OPENAPI 3.0

- ➔ OpenAPI Specification (OAS) defines a standard, language-agnostic interface to RESTful APIs
- ➔ Allows both humans and computers to discover and understand the capabilities of the service

OPENAPI 3.0

- OpenAPI Document
    - Info object
    - Servers object
    - Paths object
    - Components object
    - Security object
    - Tags object
    - externalDocs object
- Demo with OpenAPI-Delphi

NEON LIBRARY

NEON LIBRARY

- ➔ JSON serializer
  - ◆ Used mostly in REST-like scenario
- ➔ Open source Apache 2.0
- ➔ github.com/paolo-rossi/delphi-neon
- ➔ Tested on Delphi 10.x
  - ◆ Probably works also in XE7, XE8

NEON LIBRARY

- Extensive configuration: INeonConfiguration interface
  - Word case (UPPERCASE, lowercase, PascalCase, camelCase, snake_case)
  - CuStOM CAse (through anonymous method)
  - Member types (Fields, Properties)
  - Option to ignore the "F" if you serialize the fields
  - Member visibility (private, protected, public, published)

NEON LIBRARY

- Extensive configuration
  - Word case (UPPERCASE, lowercase, PascalCase, camelCase, snake_case, CuStOM CAse
  - Member types (Fields, Properties)
  - Option to ignore the "F" if you serialize the fields
  - Member visibility (private, protected, public...)
  - Custom serializer registration
  - Use UTC date in serialization
  - Pretty Printing

NEON LIBRARY

- Delphi types support
  - Basic types
    - string, Integer, Double, Boolean, TDateTime
  - Complex types
    - Arrays of (basic types, records, classes, etc...)
    - Records with fields of... anything
    - Classes with fields of... anything
    - Generic lists
    - Dictionaries (key must be of type string, enum)
    - Streamable classes

NEON LIBRARY

- ➔ Custom Serializers
  - ◆ Inherit from TCustomSerializer and register this new class in the configuration
  - ◆ Supported classes, records, arrays…
  - ◆ In the custom serializer you can continue with the standard serializer
- ➔ Demo

OPENAPI-DELPHI

OPENAPI-DELPHI

- ➔ OpenAPI 3.0 generator and parser
  - ◆ Validator is a work in progress
- ➔ Open source Apache 2.0
- ➔ github.com/paolo-rossi/OpenAPI-Delphi
- ➔ Tested on Delphi 10.x
  - ◆ Probably works also in XE7, XE8
- ➔ Designed a✎er Microso✎ OpenAPI.NET

OPENAPI-DELPHI

- ➔ Plain Old Delphi Objects as models
  - ◆ Very complex object model
  - ◆ Di✎cult in a statically typed language
  - ◆ Very di✎cult to serialize -> Neon Library

SOURCE CODE

```
procedure TForm1.Button1Click(Sender: TObject);
var
  LSchema: TOpenAPISchema;
begin
  LSchema := TOpenAPISchema.Create;

  LSchema.Title := 'Titolo';
  LSchema.Type_ := 'object';

  LSchema.Not_ := TOpenAPISchema.Create;
  LSchema.Not_.Title := 'SubSchema';

  Memo1.Lines.Text := TNeon.ObjectToJSONString(LSchema, GetNeonConfiguration);

  LSchema.Free;
end;
```

SOURCE CODE

```
procedure TForm1.Button1Click(Sender: TObject);
var
  LSchema: TOpenAPISchema;
begin
  LSchema := TOpenAPISchema.Create;
  try
    LSchema.Title := 'Titolo';
    LSchema.Type_ := 'object';

    LSchema.Not_ := TOpenAPISchema.Create;
    LSchema.Not_.Title := 'SubSchema';
    LSchema.Type_ := 'string';

    Memo1.Lines.Text := TNeon.ObjectToJSONString(LSchema, GetNeonConfig);
  finally
    LSchema.Free;
  end;
end;
```

That's all Folks!