



**DelphiDay**  
italian conference

# Da Client/Server a REST

scrivendo meno codice possibile



**PAOLO ROSSI**

**WINTECH ITALIA - CTO**

**SENCHA & EMB. MVP**



 [blog.paolorossi.net](http://blog.paolorossi.net)

 [paolo@paolorossi.net](mailto:paolo@paolorossi.net)

 [twitter.com/awebguy](https://twitter.com/awebguy)

 [github.com/paolo-rossi](https://github.com/paolo-rossi)

 [linkedin.com/in/paolo-rossi-pc](https://linkedin.com/in/paolo-rossi-pc)



20 Novembre 2024  
Padova





# GITHUB PROJECTS



[github.com/paolo-rossi](https://github.com/paolo-rossi)



## Delphi JWT

JSON Web Token Library



## WiRL

REST Library for Delphi



## Linux Daemon

Real Linux daemons



## Delphi Neon

JSON Serialization Library



## OpenAPI-Delphi

OpenAPI 3.0 Library



## NATS Delphi

NATS Client Library for Delphi



# AGENDA

---

1. Introduction
2. N-tier architecture
3. Migration
4. How to migrate the Delphi code
5. Security



# Introduction

# 1



# USE CASE SCENARIO

C/S applications	n-tier applications
LAN clients	LAN clients
	Web apps
	Mobile apps
	Automation apps
	IoT gateway/supervisor



# TYPICAL DELPHI APP

---

- C/S apps with direct access (LAN) to the database
  - ◆ A lot of BDE-based applications
- Socket communication only for devices
- HTTP communication only to interact with some web server
- SOAP clients with some third party server



# TYPICAL C/S APP

---

- App with LAN based DB connectivity
- Very rich UI (sometimes too rich!)
- UI components attached directly to design-time datasets
- A lot of events to glue it all

**The very definition of a monolith!**



# FROM C/S TO N-TIER

---

- Transition occurred around 2008-2009 for other languages
- Delphi apps for the large part are still desktop apps with direct access to the database

An it's not Delphi's fault !!

We have to fill the gap!



# N-tier Architecture

# 2



# N-TIER, WHAT FOR??

---

- Web apps
- Mobile apps
- DB-based apps (classic C/S apps)
- Automation apps
- IoT gateway/supervisor



# N-TIER, WHAT IS IT??

---

- An n-tier app is essentially an HTTP (TCP, UDP, etc...) service that listens on a port
  - ◆ Requests are from several client
  - ◆ Requests can be at the same time
  - ◆ At each request the server spawns a thread
- The data (object, variables) must be thread-safe
  - ◆ The code that access the data must be thread-safe



# Thread-Safety

# 3



# C/S APPS

---

- Usually not thread-safe at all!
- UI (VCL) it's not thread-safe by design
- Data access and classes better be thread-safe



# N-TIER APPS

---

- Server: No UI (VCL/FMX)
- Data access (and classes) **must** be thread-safe
  - ◆ DataModule, Queries, etc...
- Client: UI must synchronize with data (sometimes from different threads)
- Clients usually don't have multi-thread problems per-se



# IN DETAILS:

---

- Local variables/objects are (usually) thread-safe
- Global variables/objects are not!
- UI components are not (no need to migrate these)
- Design-time components (usually) are not!
- DataModules? it depends!
- Make routines to safely access your variables/objects
  - ◆ Learn (at least) TCriticalSection and TMonitor



# MEMORY LEAKS

---

- In REST services much more “destructive”
- First line in \*.dpr
  - ◆ `ReportMemoryLeaksOnShutdown := True;`
-



**Demo:**

## **Thread Safety**

---

- Simple Delphi DataModule
- Delphi DataSets
- Lists & Collections
- Global Objects
- Global Variables



# Migration

# 4



# MIGRATION

---

- What to migrate
  - ◆ Data access units (DataModules)
  - ◆ Business logic units (DataModules, other classes)
  - ◆ Utility classes
- Migrate or start from scratch?



# DATA ACCESS MIGRATION

---

- Still using the BDE?
  - ◆ FireDAC migration
- Not using a DataModule?
  - ◆ Please use a DataModule!
- Code is not thread-safe?
  - ◆ Make your DB code thread-safe

90% of your DB code is ready to be migrated to a service



# How to migrate code

# 5



# PROJECT DEPENDENCIES

---

- Unit (code) dependencies
  - ◆ GraphViz tools (D12)
- A single dependency (a DataModule for example) means compiling an entire project (forms, utilities, vcl, etc...)



# USES CLAUSES

---

- Use complete namespaces (fast compile time)
  - ◆ `System.SysUtils`, `Vcl.Controls`, etc...
- Order uses clauses by namespaces
  - ◆ `System.*`
  - ◆ `Data.*`
  - ◆ `Vcl.*`/`Fmx.*`
  - ◆ Libraries
  - ◆ Project's units



# USES CLAUSES

---

- Remove inter-dependencies
  - ◆ Circular references (also in implementation)
- Remove unnecessary namespaces
  - ◆ Remove all project forms from DataModules/classes
  - ◆ Remove Vcl.\* / Fmx.\* from DataModules/classes
  - ◆ Remove WinApi.\* from DataModules/classes that might be compiled in Linux/MacOS/Android/iOS
  - ◆ Keep ordered the uses clauses, move the units that Delphi keep adding to the bottom



# UNIT DEPENDENCIES

---

- If you have a circular ref between forms and a dm
  - ◆ Create a 3rd unit with the types in common



**Demo:**

## **Unit dependencies**

---

- Forms
- DataModules
- Units



# GLOBAL TO LOCAL

---

## → Variables

- ◆ Inside classes (form)
- ◆ fields or class vars

## → Constants

- ◆ Inside classes

## → Functions

- ◆ Inside classes
- ◆ Instance methods or class methods



**Demo:**

## **Global to Local**

---

- Variables
- Constants
- Functions



# GLOBAL TO LOCAL

---

- Objects (singleton)
  - ◆ Implementation reference
  - ◆ initialization/finalization
  - ◆ Lazy creation
- Static classes
  - ◆ Collection of functions (utils)
  - ◆ Collection of functionalities



**Demo:**

## **Global to Local**

---

- Objects
- Static classes



# GLOBAL SYNC

---

- If you can't go local
- Using TCriticalSection or other sync objects
  - ◆ System.SyncObjs
- Use TMonitor with objects



**Demo:**

## **Global Sync**

---

- TCriticalSection
- Other sync objects



# LISTS & COLLECTIONS

---

- Old style lists
  - ◆ TList, TObjectList, etc...
- Derived classes with method overrides
  - ◆ Sync in the override methods
- New generic classes
  - ◆ Use thread-safe classes
  - ◆ Create thread-safe classes



# LISTS & COLLECTIONS

---

- Thread-safe lists are not “magical”
  - ◆ Beware of algorithm composed of multiple methods calls
    - Looping through items and adding/deleting/changing
- Add Function that return arrays
  - ◆ For filters, sorting, etc...



**Demo:**

## **Lists & Collections**

---

- Override methods
- Filtering, sorting



# FORM TO DATAMODULE

---

- Remove any SQL string from the forms
- DataModules as data services (functions)
- Break the “only” mega DataModule in smaller ones
  - ◆ Share the connection DM
- Write thread-safe code in DataModules



**Demo:**

## **DataModules & Forms**

---

- Functions returning simple values and datasets
- Remove SQL management from Forms



# Security

# 4



# SECURITY

---

- Think about security from day 0
- Your service(s) will be accessed from outside the LAN
  - ◆ Meaning: Internet
- Never expose your database server
- Use REST libraries with known security implementations
  - ◆ Use always **JWT** as a token that contains client side information
  - ◆ Learn all about **JWT** and its use



THANK YOU