

ITDevCon  
European Delphi Conference  
14, 15 november 2013 - VERONA (Italy)

delphimvcframework  
RESTful interfaces and dynamic  
web pages with Delphi

Daniele Teti  
R&D Director & Educational

bit Time software

## What is DelphiMVCFramework?

ITDevCon  
European Delphi Conference

- Powerful XE3, XE4 and XE5 framework for web solutions
- Built in WebServer
- HTTP/HTTPS support
- It use Embarcadero WebBroker but is not tied to it
- Inspired to Java JAX-RS
- It is simple
- You can be productive in few minutes

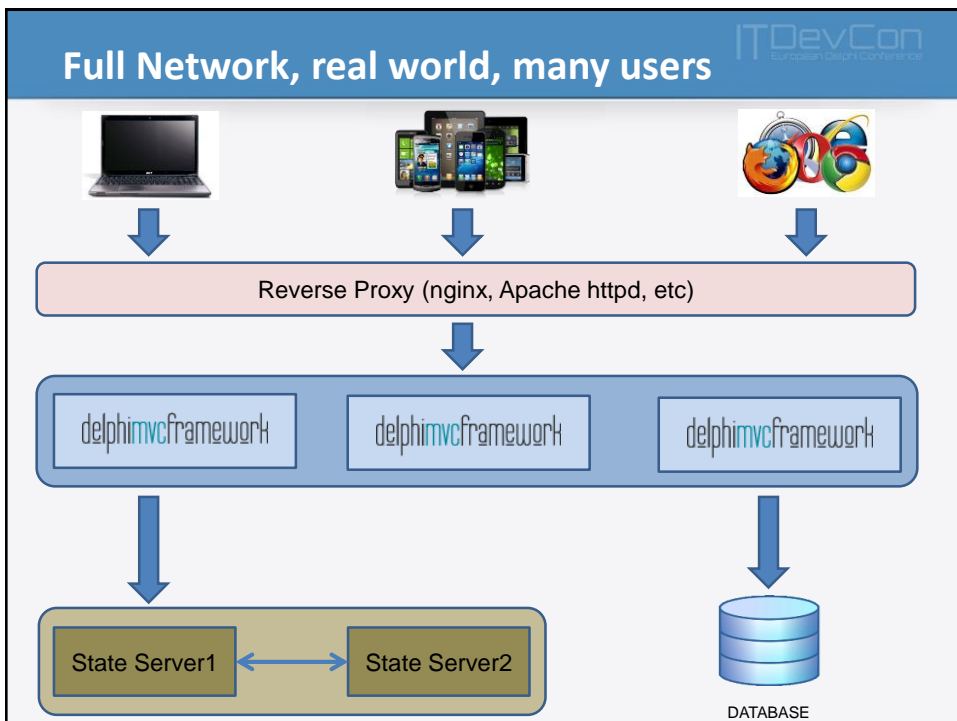
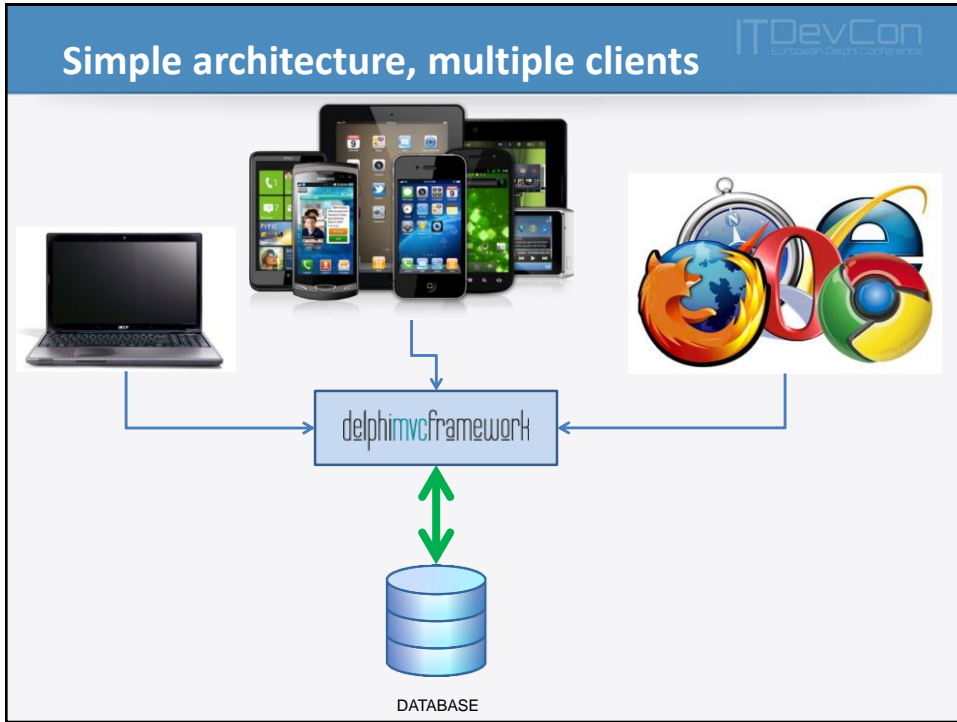
delphimvcframework

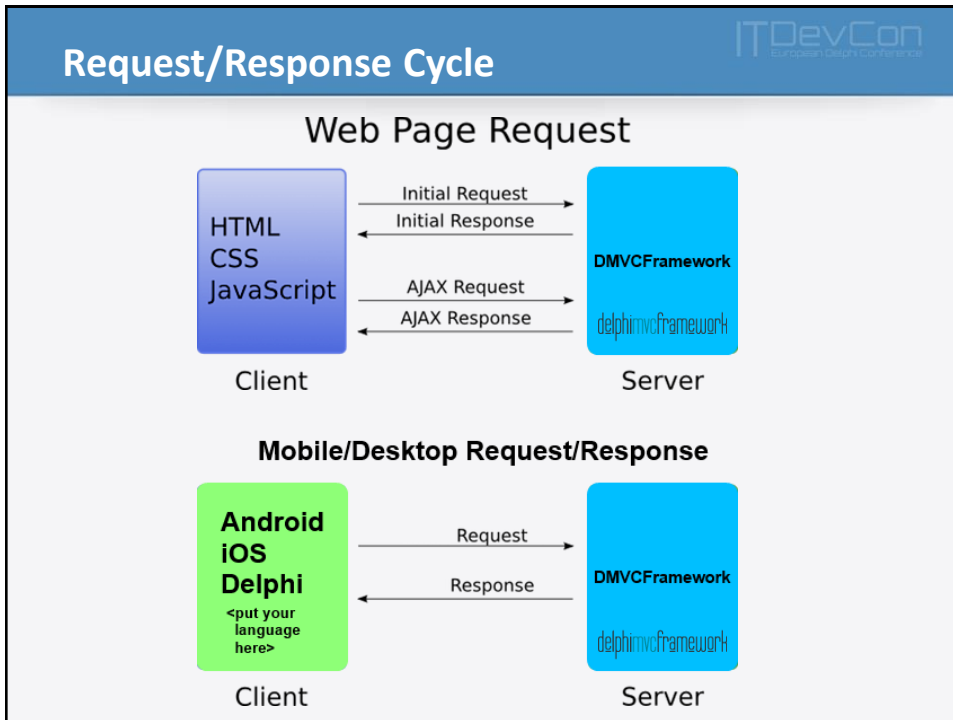
## DMVCFramework main features

- RESTful
  - Richardson Maturity Model Level 3
- Designed with services and web client app in mind
- Server side generated pages using eLua (Embedded Lua)
- Can be used in load balanced environment using memcached (memcached.org)
- Fancy URL with parameter mappings
- Integrated Delphi RESTClient
- Messaging extension using STOMP
- Experimental support for IOCP

## What kind of applications can be build with DMVCFramework?

- Application Servers
- RESTful web services
- Classic web application
- Web Client Applications
- Messaging solutions
  - based on Apache ActiveMQ or Apache Apollo
- Delphi thin clients
- Mobile and Web backends
- Scalable (Load balanced) web systems
- Secure servers with HTTPS





## Your first DMVCFramework

ITDevCon  
European Delphi Conference

```

type
  TWebModule1 = class(TWebModule)
    procedure WebModuleCreate(Sender: TObject);
  private
    DMVC: TMVCEngine;
  end;

. . .

procedure TWebModule1.WebModuleCreate(
    Sender: TObject);
begin
  DMVC := TMVCEngine.Create(self);
end;
  
```

# Richardson Maturity Model Level 3

ITDevCon  
European Delphi Conference

The diagram illustrates the Richardson Maturity Model with four levels represented by horizontal bars of increasing width and height from bottom-left to top-right. A large vertical arrow on the right points upwards, and a haloed circle is positioned at the top right. A box labeled 'DMVCFramework' has an arrow pointing to 'Glory of REST'.

- Level 0: The Swamp of POX
- Level 1: Resources
- Level 2: HTTP Verbs
- Level 3: Hypermedia Controls

DMVCFramework → Glory of REST

<http://martinfowler.com/articles/richardsonMaturityModel.html>

# ITDevCon

European Delphi Conference

Abstract graphic consisting of several overlapping, semi-transparent rings in shades of blue, green, orange, and yellow, with a small globe icon in the center.

## Controllers, Actions and Routing

DelphiMVCFramework

## Architecture of DMVCFramework Server

ITDevCon  
European Design Conference

- One Application
- Many Controllers
  - Classes inherited from TMVController
- Many Actions for each controller
  - Controller Actions are its methods instrumented with a specific attribute
- Controller is addresses using a piece of URL
- Actions are selected using the second part of the URL

## Architecture of DMVCFramework Server

ITDevCon  
European Design Conference

- One Application
- Many Controllers
  - Classes inherited from TMVController
- Many Actions for each controller
  - Controller Actions are its methods instrumented with a specific attribute
- Controller is addresses using a piece of URL
- Actions are selected using the second part of the URL

Server Name            Controller            Action  
www.myserver.com/people/rome/danieleteti

## Controllers and Routing



**type**

```
[MVCPATH('/blog')]
TBlog = class(TMVCController)
public
  [MVCPATH('/posts/($year)/($month)/($title)')]
  procedure GetArticle(CTX: TWebContext);
end;
```

Matching URI Samples

```
/blog/posts/2011/05/a-brand-new-framework
/blog/posts/2013/05/rest-rest-for-delphi
/blog/posts/1/5/thisAndThat
/blog/posts/2013/05/123
```

## DMVCFramework Attributes



Name	Scope	Optional	Used for routing?
MVCPATH	Class Method	No	Yes
MVCHTTPMethod	Method	Yes	Yes
MVCConsumes	Method	Yes	Yes
MVCProduces	Method	Yes	No

## Controllers and Routing (1)



```
[MVCPath('/blog')]
TBlog = class(TMVCController)
public
  [MVCHttpMethod([httpGET])]
  [MVCPath('/posts/($year)/($month)/($title)')]
  procedure GetArticle(CTX: TWebContext);

  [MVCHttpMethod([httpPOST])]
  [MVCPath('/posts/($year)/($month)/($title)')]
  procedure CreateArticle(CTX: TWebContext);
end;
```

## Controllers and Routing (2)



```
[MVCPath('/blog')]
TBlog = class(TMVCController)
public
  [MVCHttpMethod([httpDelete])]
  [MVCPath('/posts/($year)/($month)/($title)')]
  procedure DeleteArticle(CTX: TWebContext);

  [MVCHttpMethod([httpPOST, httpPUT])]
  [MVCPath('/posts/($year)/($month)/($title)')]
  procedure UpdateArticle(CTX: TWebContext);
end;
```



## Controllers and Routing (3)



```
[MVCPath('/blog')]  
TBlog = class(TMVCController)  
public  
    [MVCProduce('application/json', UTF8)]  
    [MVCHTTPMethod([httpGET])]  
    [MVCPath('/posts/($year)/($month)')]  
    procedure GetArticleByMonth(  
        CTX: TWebContext);  
end;
```

## Controllers and Routing (4)



```
[MVCPath('/blog')]  
TBlog = class(TMVCController)  
public  
    [MVCProduce('application/json', UTF8)]  
    [MVCConsumes('application/json')]  
    [MVCHTTPMethod([httpPOST])]  
    [MVCPath('/posts/($year)/($month)')]  
    procedure CreateArticle(  
        CTX: TWebContext);  
end;
```

## Request parameters



- Query String
  - GET, POST, PUT, DELETE, HEAD, OPTIONS
- URL Mapped
  - GET, POST, PUT, DELETE, HEAD, OPTIONS
- Request Body
  - POST, PUT, OPTIONS
- Cookies
  - GET, POST, PUT, DELETE, HEAD, OPTIONS

## HTTP Safe Methods



- The convention has been established that the GET and HEAD methods SHOULD NOT have the significance of taking an action other than retrieval.
- These methods ought to be considered "safe".
  - It is not possible to ensure that the server does not generate side-effects as a result of performing a GET request (eg. Logging, audit, tracing). However, the important distinction here is that the user did not request the side-effects, so therefore cannot be held accountable for them.

## HTTP Idempotent Methods

ITDevCon  
European Design Conference

- With IDEMPOTENT methods the side-effects of  $N > 0$  identical requests is the same as for a single request.
- The methods GET, HEAD, PUT and DELETE share this property. Also, the methods OPTIONS and TRACE SHOULD NOT have side effects, and so are inherently idempotent.

## Reading Parameters

ITDevCon  
European Design Conference

`Context.Request.Params[ 'ParamName' ]`

Reads Parameters in the following order

- URL Mapped parameters
- Query String parameters
- FORM parameters (eg. HTML Form Submit)
- Cookie fields

## URL mapped parameters



**GET /blog/posts/danieleteti/2013/11**

```
[MVCPath('/posts/($user)/($year)/($month)')]
[MVCHTTPMethod([httpGET])]
procedure GetArticles (CTX: TWebContext);
. . .
procedure GetArticles (CTX: TWebContext);
var
    year,month: Integer; user: String;
begin
    user := CTX.Request.Params['user'];
    year := CTX.Request.Params['year'].ToInteger;
    month := CTX.Request.ParamsAsInteger['month'];
end
```

## QueryString mapped parameters



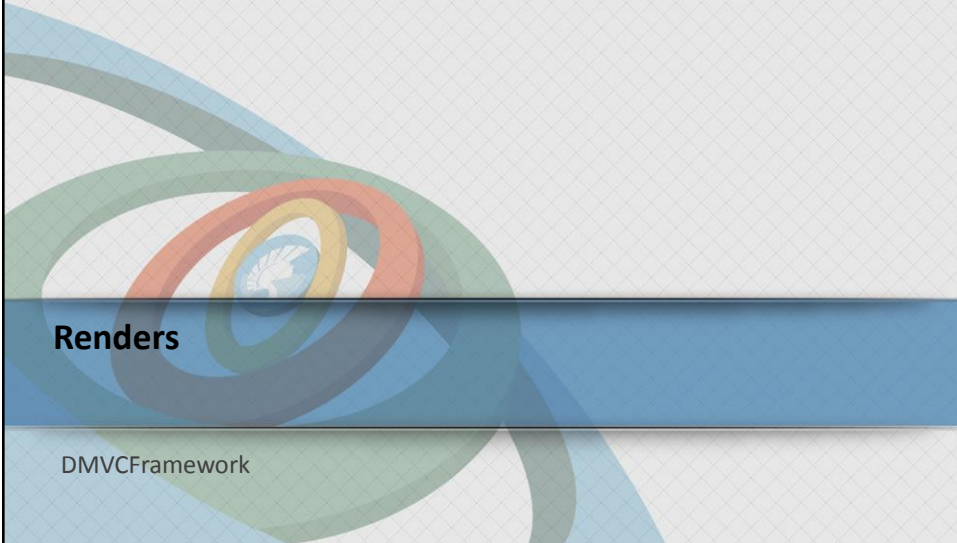
**GET /blog/posts/danieleteti?year=2013&month=11**

```
[MVCPath('/posts/($user)')]
[MVCHTTPMethod([httpGET])]
procedure GetArticles (CTX: TWebContext);
. . .
procedure GetArticles (CTX: TWebContext);
var
    year,month: Integer; user: String;
begin
    user := CTX.Request.Params['user'];
    year := CTX.Request.Params['year'].ToInteger;
    month := CTX.Request.Params['month'].AsInteger;
end
```

**DEMO** ITDevCon  
European Delphi Conference

- Routing

ITDevCon  
European Delphi Conference



**Renders**

DMVCFramework

## Using Renders

- «Renders» are a set of controller methods that actually produces the response stream
- There are a lot of Render methods with overload for many types of data
- Renders can set the HTTP Status Code and the ContentType
- Currently you can render
  - Text
  - DataSets
  - JSONValues
  - Delphi Objects
  - List of Delphi Objects
  - Exceptions
  - HTML pages using eLua
  - Raw Streams

## Render a TObject

```
procedure TPeople.GetPerson(CTX: TWebContext);
var
  P: TPerson;
begin
  P := TPerson.Create;
  P.FirstName := 'Daniele';
  P.LastName := 'Teti';
  P.DOB := EncodeDate(1975, 5, 2);
  P.Married := True;
  Render(P);
end;
```

## Render a TJSONValue



```

procedure TPeople.GetPerson(CTX: TWebContext);
var
    J: TJSONObject;
begin
    J := TJSONObject.Create;
    J.AddPair('FirstName', 'Daniele');
    J.AddPair('LastName', 'Teti');
    J.AddPair('DOB',
        ISODateToString(EncodeDate(1975, 5, 2)));
    J.AddPair('Married', TJSONTrue.Create);
    Render(J);
end;

```

## Render a TDataSet



```

procedure TCustomersController.GetAll(
    CTX: TWebContext);
var
    wm: TWebModule1; //the main WebModule
begin
    wm := GetCurrentWebModule as TWebModule1;
    wm.qryCustomers.Open;
    //dataset is rendered as json array of objects
    Render(wm.qryCustomers);
end;

```

## DEMO

ITDevCon  
European Design Conference

- Renders

## Filtering Action Requests

ITDevCon  
European Design Conference

- Each action call (aka controller method) can intercepted and handled
- Controllers initialization methods are also available
  
- DEMO
  - ActionsFilter



## Session and Application Session

ITDevCon  
European Design Conference

- DMVCFramework supports session for each request
- Session is a key/value structure private for each user
- Application Session is similar but it is shared to all users (WARNING!!!)
- Session is alive for Config[ 'session\_timeout' ] minutes

```

procedure TTestServerController.Login(ctx: TWebContext);
begin
    Session['username'] := ctx.Request.Params['username'];
end;

procedure TTestServerController.Logout(ctx: TWebContext);
begin
    //destroy session with
    SessionStop(false);
end;

```

## Server Side Views with eLua

ITDevCon  
European Design Conference

- Server side views uses the Lua language just like PHP pages use PHP (code into text)
- eLua (Embedded Lua) has been specifically designed for DMVCFramework in its brother project called LuaDelphiBinding
- It is similar to PHP for HTML, JSP for Java and erb for Ruby

```

<?lua= 3*8 ?>           EXPRESSION
<?lua arbitrary_lua_code ?> SCRIPTLET

```

## Why Lua?

- Lua is small, fast and simple
  - The whole implementation is less than 6000 lines of ANSI C
  - Requires only one dll on Windows
- Has been designed as extension language
- Dynamic and not strongly typed
- Clear, simple and familiar syntax
- Great extensibility
- Functions as first-class values
- Tables a.k.a Associative arrays
- Garbage collection
- Allows extension of the semantics of the language.
- Widely used even outside the Delphi world

## Is Lua popular?

- Lua is the **most popular scripting language** for game programming
- **Adobe Photoshop Lightroom** uses Lua for its user interface.
- **Apache HTTP Server** can use Lua anywhere in the request process
- **Cisco** uses Lua to implement Dynamic Access Policies within the Adaptive Security Appliance.
- **Damn Small Linux** uses Lua to provide desktop-friendly interfaces
- **FreePOPs**, an extensible mail proxy, uses Lua to power its web front-end.
- **MySQL Workbench** uses Lua for its extensions & add-ons.
- **Nginx** has a powerful embedded Lua module that provides an API
- **nmap** network security scanner uses Lua as the basis for its scripting language
- **Vim** has Lua scripting support
- **VLC** media player uses Lua to provide scripting support.
- Since March 2013, Lua is used as a new template scripting language on **Wikipedia** and other Wikimedia Foundation wikis.
- **WinGate** proxy server allows event processing and policy to execute lua scripts with access to internal WinGate objects.
- **Wireshark** network packet analyzer allows protocol dissectors and post-dissector taps to be written in Lua

## eLua sample (expressions and code)

```
This is text but this is <?lua= "Lua" ?>
And now, Good <?lua
    if isMorning() then
?>
Morning
<?lua
    else
?>
Evening
<?lua
    end
?> to all the people
```

## Getting Started with eLua

- eLua files in the document\_root are executed directly
- Actions can use eLua views using LoadView('viewname')
- Lua Helpers helps to create html page faster
- Similar to the RubyOnRails form helpers

## DEMO

ITDevCon  
European Design Conference

- Simplewebapplication

## Serve static files

ITDevCon  
European Design Conference

- Use the «document\_root» to serve static files
  - HTML, Images, javascripts, binary...
- This allows to use Web Client javascript framework
  - jQuery
  - AngularJS
  - Ember.js
  - Backbone.js
  - etc

## Web Client App DEMOS

ITDevCon  
European Design Conference

- WineCellarServer
- WineCellarServerWITHDORM
- AngularJS\WebClientSample

## Embedded Logging system

ITDevCon  
European Design Conference

```
procedure Log(AMessage: string);
procedure LogW(AMessage: string);
procedure LogE(AMessage: string);
procedure LogEx(
    AException: Exception;
    AMessage: string);
procedure Log(LogLevel: TLogLevel;
    const AMessage: string);
procedure LogEnterMethod(AMethodName: string);
procedure LogExitMethod(AMethodName: string);
```

Configurable LogLevel using Config

## Load Balancing

ITDevCon  
European Design Conference

- DMVCFramework has been designed with clustering in mind
- Session factory allows 2 kind of sessions
  - Memory
    - Do not support load balancing
    - Sessions do not survives to restart
  - State Server
    - Support load balancing
    - Sessions survive to restart
    - Require a memcached daemon

## Messaging estensions

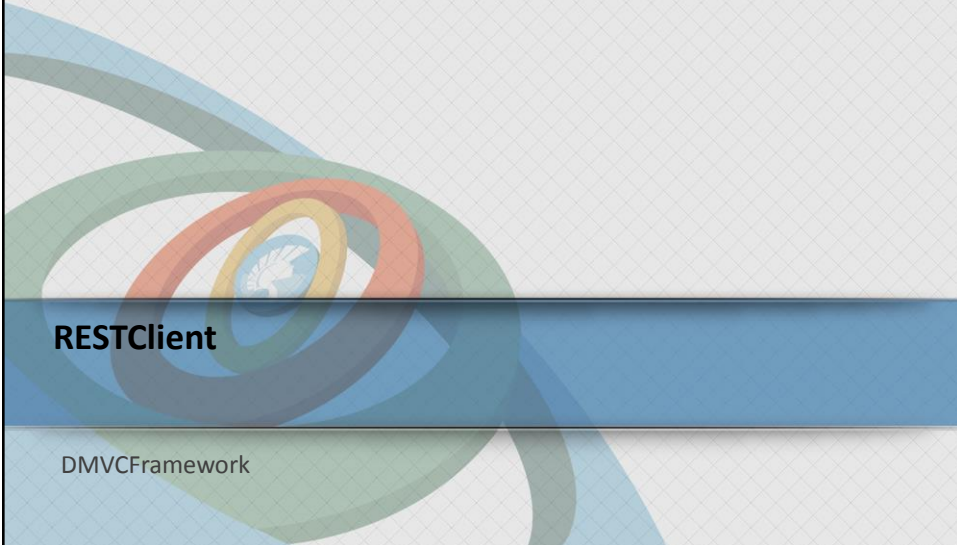
ITDevCon  
European Design Conference

- A specific system controller support Messaging Extensions
- TMVCBUSController mapped to /messages
- Provides the following routes
  - `[MVCPATH('/subscribe/($name)')]`
  - `[MVCPATH('/unsubscribe/($name)')]`
  - `[MVCPATH('/receive')]`
  - `[MVCPATH('/enqueue/($topic)')]`
  - `[MVCPATH('/topics')]`
- Method to enqueue message available
  - `EnqueueMessageOnTopic(topicname, jsonobject);`

**Messaging Extensions DEMO** ITDevCon  
European Delphi Conference

- CallbackDemo

ITDevCon  
European Delphi Conference



**RESTClient**

DMVCFramework

## RESTClient



- Simple Delphi client for DMVCFramework
- Completely integrated with the **Mapper**

```

var
  rest: TRESTClient;
  response: IRESTResponse;
  Person: TPerson;
begin
  rest := TRESTClient.Create('localhost', 3000);
  response := rest.doGET('/people', ['1']);
  Person := Mapper.
    JSONObjectToObject<TPerson>(response.BodyAsJSONObject);
  ShowMessage(Person.FullName);
  Person.Free;
  rest.Free;
end;

```

## RESTClient Asynch



```

RESTClient.Asynch(
  procedure(Response: IRESTResponse)
  begin
    //do something with response
  end,
  procedure(E: Exception)
  begin
    //do something with exception
  end).
doPOST('/echo', ['one', 'two'],
  'Hello World');

```



## RESTClient Async

ITDevCon  
European Delphi Conference

```
RESTClient.Async(  
  procedure(Response: IRESTResponse) begin  
    //do something with response  
  end,  
  procedure(E: Exception) begin  
    //do something with exception  
  end,  
  procedure begin  
    //when finished (success or failure)  
  end).  
doPOST('/echo', ['one', 'two'],  
  'Hello World');
```

ITDevCon  
European Delphi Conference

DMVCFramework SubProjects

DMVCFramework

## Mapper

- Powerful converter for Delphi Objects, JSONValues and DataSets
- Some notable conversions
  - ObjectToJSONObject
  - JSONObjectToObject<T>
  - ObjectListToJSONArray
  - JSONArrayToObjectListof<T>
  - DataSetToJSONArray
  - JSONArrayToObjectListof<T>

## LuaDelphiBinding

- Lua bind for Delphi
- Uses Lua 5.1
- Helper methods to push Delphi dictionaries and Delphi Object as Lua Tables
- Helper to publish functions
- **LuaTextFilter** to handle **eLua** even outside the DMVCFramework

The image is a thank-you slide for the ITDevCon European Delphi Conference. It features a light gray background with a subtle grid pattern. On the left side, there is a graphic of three overlapping, 3D-style rings in green, orange, and yellow, with a blue circular icon containing a white bird-like symbol in the center. The text 'ITDevCon' is prominently displayed at the top in a large, blue, sans-serif font, with 'IT' in a lighter blue and 'DevCon' in a darker blue. Below it, 'European Delphi Conference' is written in a smaller, blue font. The dates '14, 15 november 2013 - VERONA (Italy)' are centered below the title. To the right, the words 'Thank You' are written in a large, black, sans-serif font. In the bottom right corner, the contact information for Daniele Teti is listed: 'Daniele Teti', 'd.teti@bittime.it', 'www.danieleteti.it', and '@danieleteti'. At the very bottom right, the 'bit Time software' logo is visible, consisting of the text 'bit Time' in a blue font with 'software' in a smaller font above it, and a small blue icon to the right.

# ITDevCon

European Delphi Conference

14, 15 november 2013 - VERONA (Italy)

## Thank You

Daniele Teti  
[d.teti@bittime.it](mailto:d.teti@bittime.it)  
[www.danieleteti.it](http://www.danieleteti.it)  
@danieleteti

bit Time software