31ST EMBARCADERO DEVELOPER CAMP

第 31 回　エンバカデロ・デベロッパーキャンプ

【T4】Delphi/C++テクニカルセッション

「「カラオケの鉄人」店舗で、クーポン作成 ―DTP並みの印刷システム実現の手法」

株式会社鉄人化計画
新規事業開発部　毛利春幸

embarcadero

「「カラオケの鉄人」店舗で、クーポン作成―DTP並みの印刷システム実現の手法」

はじめに

はじめに

- 株式会社鉄人化計画では、「カラオケの鉄人」の各店舗で、ニーズに合わせてチラシやクーポンを即時発行・印刷でいるシステムを開発しました。Webベースのこのシステムは、RAD StudioのWebBrokerとFireMonkey、更にjQueryを活用し、DTP並みの印刷を実現しています。

「カラオケの鉄人」と他店の違い(1)

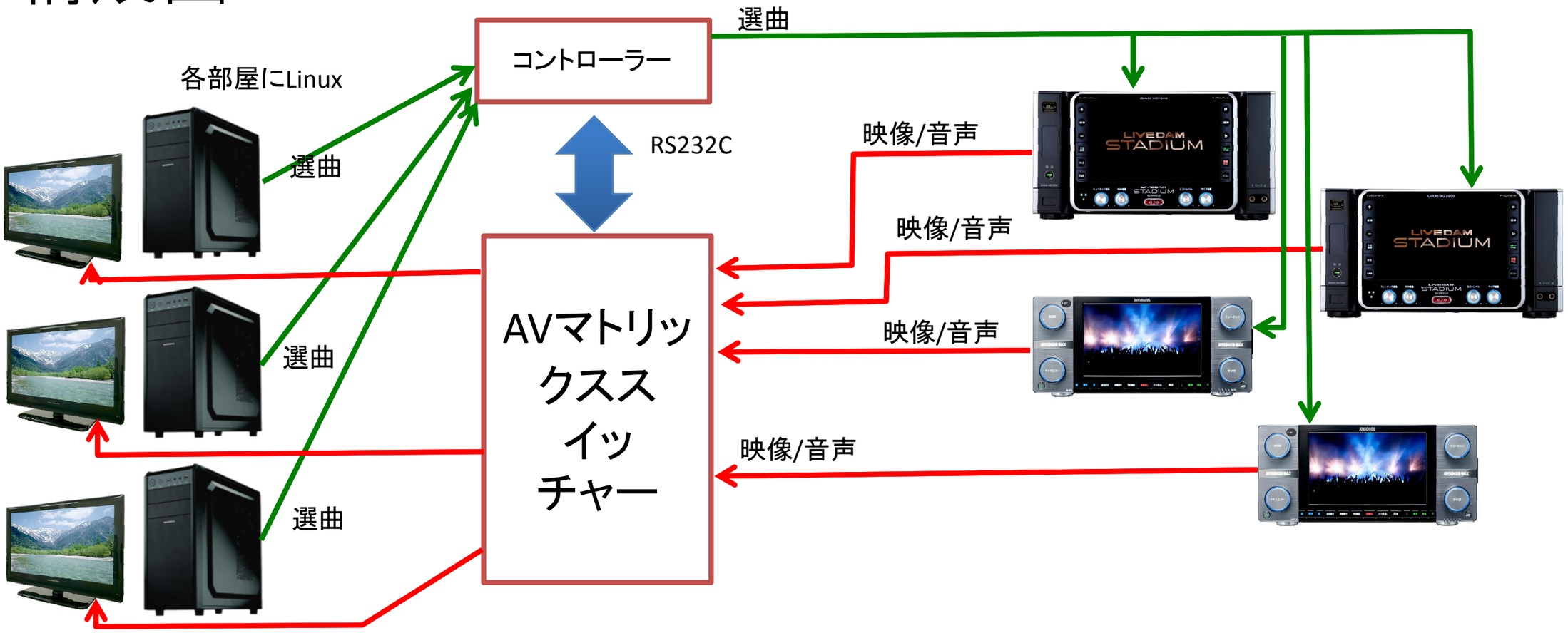
- 集中管理カラオケシステムです
- メーカのカラオケ再生機が部屋に無い

←部屋に置いてない

- カラオケ再生機は機械室にありAVをマトリックススイッチャーを使って部屋まで転送させる仕組みです。(今はデジタルデータを転送しています)

「カラオケの鉄人」と他店の違い(2)

- 構成図



鉄人システムとボーランド

- 鉄人システムは2002年ごろ完成
- 当時全ての部屋端末はLinux + Kylixの構成
- 今でも1000部屋近くLinux + Kylixの構成です
- 2008年ごろ会員システム導入 全てDelphi WebBroker です
- 今年 鉄人化計画とシステム開発会社であった、システムプランベネックスが吸収合併しました

なぜKylix

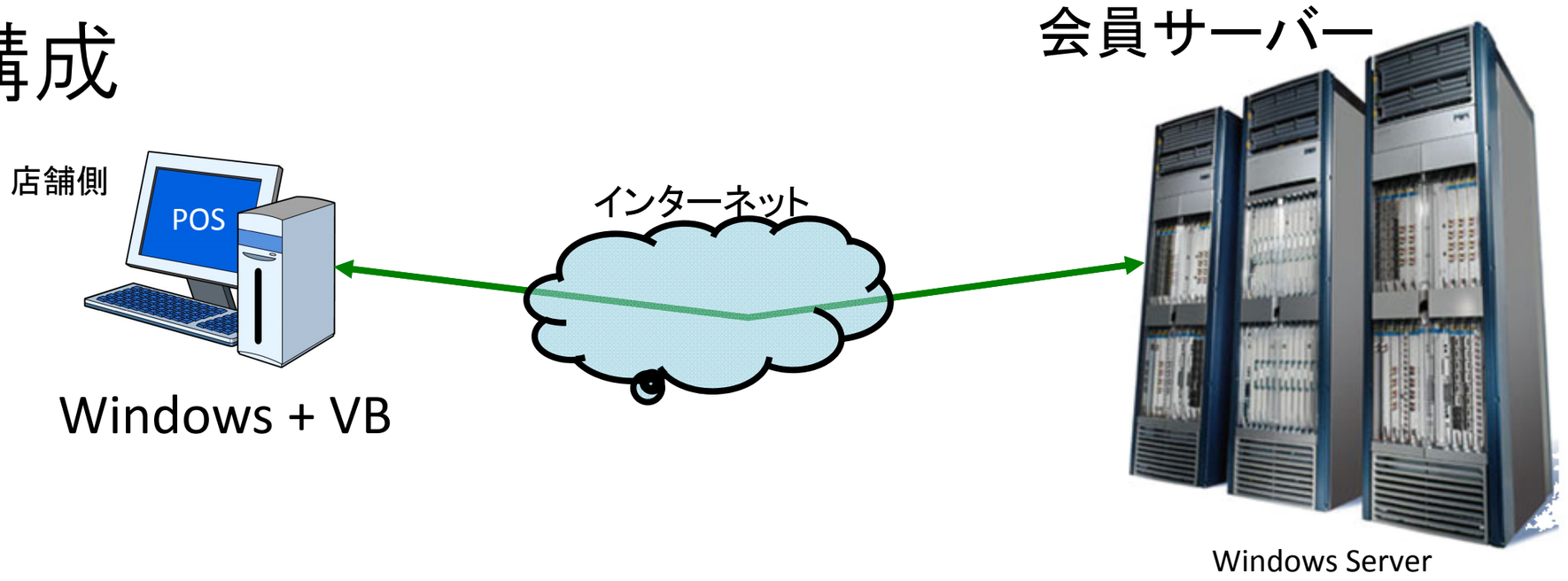
- 2001～2002年ごろLinuxがブームでした。
- 部屋側の端末を1円でも安くあげたい
  OSの費用もかけたくない
- 当時流行ってたGTK+やQtと比較してKylixのIDEは視覚的に理解しやすく開発しやすかった

なぜ会員システムはWebAPI(1)

- 店舗側からの接続がインターネット経由でした
- POS側で使われていたSQL Server+ Windows+VBでした
- 会員システムも同じ構成をとる事が出来た
- IIS+(ISAPI and CGI)構成が取れた

なぜ会員システムはWebAPI(2)

- 構成



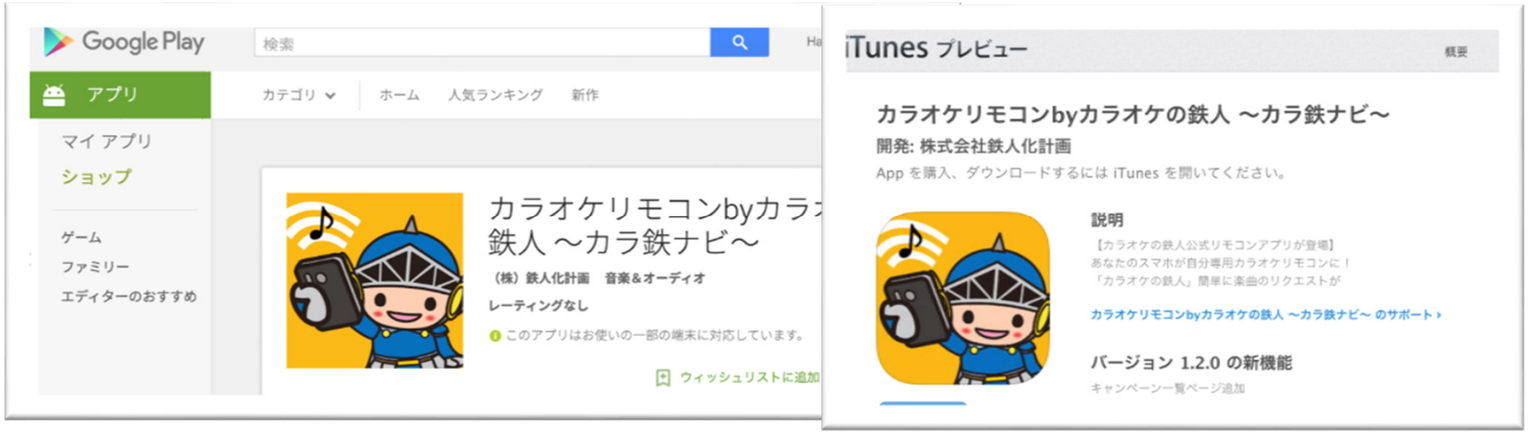
集中管理カラオケ開発者側から見たメリット

- カラオケ機器が機械室にある為効率の良い再生管理ができます
- フロント(受付)清算POSとの連動
- お客様が歌われた歌唱履歴が取れます
- 部屋にLinux 端末があるので機能が拡張できる

部屋の拡張

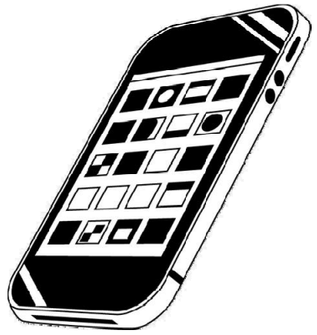
- 電子目次本 カラ鉄ナビの導入

- スマホアプリ　カラ鉄ナビアプリ導入



部屋にLinux + Kylix

- 部屋にLinux + Kylixがあるので 楽曲の再生管理や楽曲リクエストがスマホなどでできます



Wi-Fi経由で部屋端末に接続し
楽曲をリクエストできます

Kylix

カラオケ映像

カラオケの鉄人

- カラオケの店舗を運営してる会社なのですが思いついたシステムは社内で作ってしまう文化が根付いていました

- 他店とは違い店舗側(社員)は、
わりとITに詳しい

今回の案件

- クーポン システム
  (入力、(1タイム/期間限定)コード発行、POS連動)

- クーポン券 印刷カスタマイズするシステム
  (店舗側で印刷)

社内の状況(問題点)

- むやみに実行ファイルをインストールできない。
  (EXEファイルの配布できるのですが
  いろいろ手順踏まないといけないので時間もかかります。)
- 今回の案件は 年末商戦で利用する為上記の手順を踏むと年明ける

状況からの結論

- Webしか無いかと

構成図

本部からクーポン発行

メインWeb画面は jQuery

Linuxサーバ

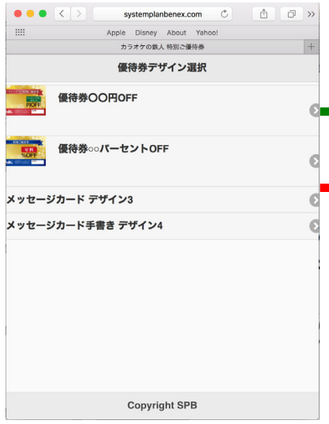
Windowsサーバ

Apache+
php

SQL Server
クーポンDB

店舗側

優待券印刷Web画面



メインWeb画面はjQuery

画像生成依頼

優待券画像完成データダウンロード

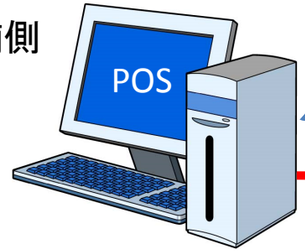
Ajax JSON + WebBroker

Windowsサーバ

IIS+
Delphi

ADO Query問合せ

店舗側



優待券を来店時にフロントバーコードリーダーで読み取る

POSレジがバーコードが正しいかWebBrokerに問合せる

2つの案件なぜ来たのか

- クーポンシステムに関しては以前 昨年某S社様が営業に来られて そのアイデアを鉄人風にアレンジ

- 印刷システムに関しては(次頁)

3月商戦時の画像作成システムをアレンジ

- ある日(3月)社内メーリスで
「誰かこの画像に社員番号入れたら、自動的に優待券の画像を生成するWebアプリつくれないですか？マッハでできる人いませんか？」
- DelphiならWebサーバーに設置するまで
2時間あればできますよね

その時に作った優待券

- メーリスで書いてたようにPNG画像に社員番号を入れるだけのシステムでした

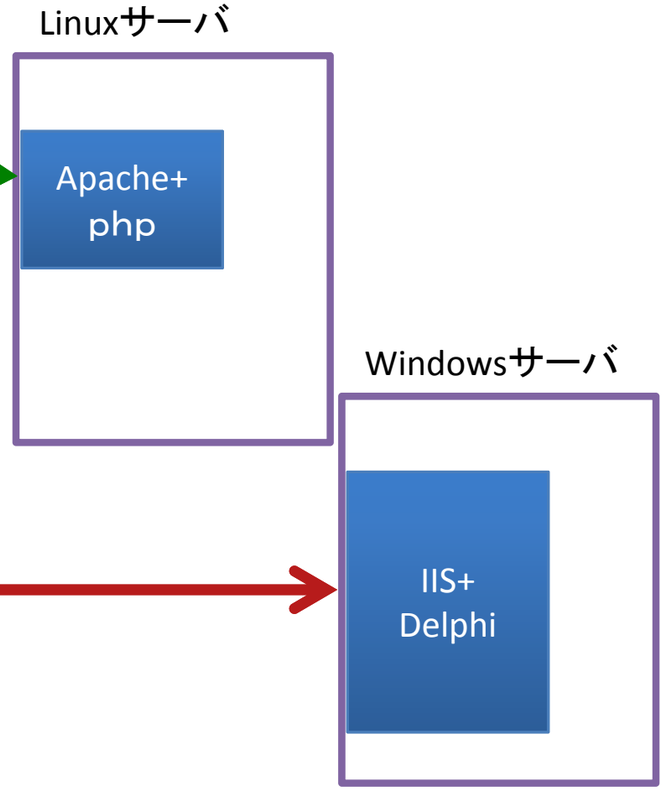
WebBroker

- 2008年ごろ弊社データセンター側システム運用を開始したのですがシステム全体の9割が現在のところIIS + WebBrokerです

- なぜ　WebBrokerなのか
- いつからあるのか忘れたのですがDelphi3ごろ?から気がつけばすでに存在していた
- 枯れた技術なので失敗はない

フロント側Web画面はjQuery+PHP

- データやメソッドはJSONでWebBrokerを使う



なぜ画面はjQueryなのか？
jQueryはクライアントブラウザ側で実行される為
サーバーサイドでプログラム実行する必要が無い。

データ取得などのやりとりだけサーバーサイドで
動的プログラムで取得する

WebBroker基本

- おさらい的に
- TWebModuleで継承されたクラスのイベントにプログラムを書きます。
- デフォルトでは

```
Response.Content :=
   '<html>' +
   '<head><title>Web サーバー アプリケーション<title></head>' +
   '<body>Web サーバー アプリケーション</body>' +    '</html>';
```

こんな事になってますが最近ブラウザはUTF8デフォなので文字化けます

WebBroker UTF-8で出力

- Response.SendResponse を使うとうまくいきますね

```
procedure TWebModule1.WebModuleDefaultHandlerAction(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
const
  DEF_STR = '<html>' +
    '<head><title>Web サーバー アプリケーション</title></head>' +
    '<body>Web サーバー アプリケーション</body>' +
    '</html>';
begin
  Fss  := TStringStream.Create(DEF_STR, TEncoding.UTF8);
  try
    Fss.Position := 0;
    Response.ContentType  := 'text/html; charset=utf-8';
    Response.ContentStream  := Fss;
    Response.SendResponse;
  except
  end;
end;

procedure TWebModule1.WebModuleDestroy(Sender: TObject);
begin
  if Assigned(Fss) then
    Fss.Free;
end;
```

WebBroker JSON出力する

- 先ほどのUTF8出力方法と同じ方法でJSONを出力するとUTF8なのでjQueryで取り込む事できます。
- ClientDataSet1に100レコードほど入れたデータをJSON出力します

```
  if ClientDataSet1.Active then
  begin
    ClientDataSet1.EmptyDataSet;
    ClientDataSet1.Active := False;
    ClientDataSet1.Fields.Clear;
  end;

  ClientDataSet1.FieldDefs.Add('num', TFieldType.ftInteger, 0);
  ClientDataSet1.FieldDefs.Add('str', TFieldType.ftString, 100);

  ClientDataSet1.CreateDataSet;
  ClientDataSet1.Close;

  ClientDataSet1.Open;
  for i := 0 to 99 do
  begin
    ClientDataSet1.AppendRecord([i + 1, Format('テスト%d', [i+1])]);
  end;
  ClientDataSet1.CheckBrowseMode;
  ClientDataSet1.Close;
end;
```

WebBroker JSON出力する(2)

- ClientDataSet1に作成したレコードを
  TJSONObjectにコピーします。

```
  ja: TJSONArray; jo: TJSONObject;
begin
  ClientDataSet1.Active := Tr
  ja  := TJSONArray.Create;
  while not ClientDataSet1.Ec
  begin
    jo  := TJSONObject.Create
    jo.AddPair('num',
      TJSONNumber.Create(Clie
      );
    jo.AddPair('str',
      TJSONString.Create(Clie
      );
    ja.Add(jo);

    ClientDataSet1.Next;
  end;
  ClientDataSet1.Active := Fa
  FJsonOut.AddPair('ClientDat
  Fss  := TStringStream.Creat
```

127.0.0.1

127.0.0.1:8080

{"ClientDataSet1":[{"num":1,"str":"テスト1"},{"num":2,"str":"テスト2"},
{"num":3,"str":"テスト3"},{"num":4,"str":"テスト4"},{"num":5,"str":"テスト5"},
{"num":6,"str":"テスト6"},{"num":7,"str":"テスト7"},{"num":8,"str":"テスト8"},
{"num":9,"str":"テスト9"},{"num":10,"str":"テスト10"},{"num":11,"str":"テスト
11"},{"num":12,"str":"テスト12"},{"num":13,"str":"テスト13"},
{"num":14,"str":"テスト14"},{"num":15,"str":"テスト15"},{"num":16,"str":"テス
ト16"},{"num":17,"str":"テスト17"},{"num":18,"str":"テスト18"},
{"num":19,"str":"テスト19"},{"num":20,"str":"テスト20"},{"num":21,"str":"テス
ト21"},{"num":22,"str":"テスト22"},{"num":23,"str":"テスト23"},
{"num":24,"str":"テスト24"},{"num":25,"str":"テスト25"},{"num":26,"str":"テス
ト26"},{"num":27,"str":"テスト27"},{"num":28,"str":"テスト28"},
{"num":29,"str":"テスト29"},{"num":30,"str":"テスト30"},{"num":31,"str":"テス
ト31"},{"num":32,"str":"テスト32"},{"num":33,"str":"テスト33"},
{"num":34,"str":"テスト34"},{"num":35,"str":"テスト35"},{"num":36,"str":"テス

JSONデータをjQueryで呼びます(1)

- 先ほどDelphiで作成したJSONデータをjQueryから呼び表示させます
- クロスドメインなAjaxでJSONを扱う場合

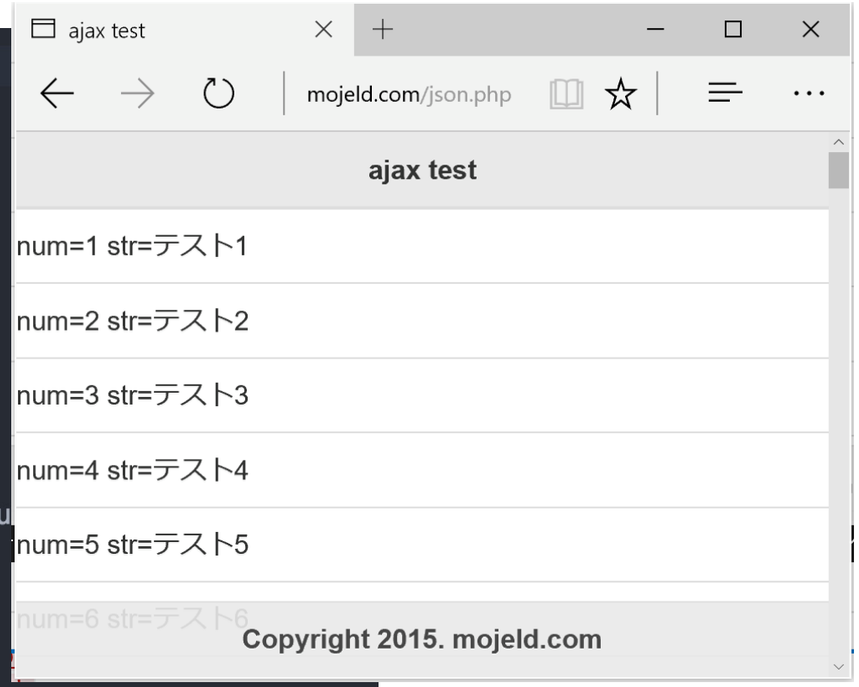
```
Response.SetCustomHeader('Access-Control-Allow-Origin','*');
Response.SetCustomHeader('Access-Control-Allow-Headers','Origin, X-Requested-With, Content-Type, Accept');
```

上記のお作法が必要です。

JSONデータをjQueryで呼びます(2)

- $(document).on(“pageinit”, function(){
  ページ読み込み時にJSONデータを取得します

```
18  function getAjaxJson(){
19    var j1 = null;
20    $.ajax({
21      type: 'POST',
22      url: 'http://10.211.55.3:8080/',
23      dataType: 'json',
24      data: {},
25      success: function(gjson){
26        $('#json_test').empty();
27        j1 = gjson.ClientDataSet1;
28        for(var i1=0; i1 < j1.length; i1++){
29             $('#json_test').append('<li >num=' + j1[i1].nu
30        }
31        $('#json_test').listview('refresh');
32      }
33    });
34  }
```



JSONP（JSON with padding）

- $.ajax()の dataType: ‘jsonp’指定すると
  JSONPのデータが取得できます。
- データ渡すJSON側の処理は関数()でくくるだけです。

```
TStringStream.Create(Format(‘hoge(%s)’, [FJsonOut.ToString]), TEncoding.UTF8);
```

- Delphiで作る場合これで良いのかと思います。

jQueryとJSONを使ってクーポン管理

- jQueryとDelphi(WebBroker)のJSONを使ってクーポンの管理画面を作りました。

- 印刷用の画像ファイルを作る方法も同じくDelphi(WebBroker)で構築する事にしました。

JSON出力をBSONに移行(1)

- 10 Seattle からBSONが使えます
  BSONは主にMongoDBのデータストレージ及びネットワーク転送フォーマットとして利用されている、データ交換フォーマットである。単純なデータ構造や連想配列（MongoDBではオブジェクトまたはドキュメントと表す）を示すバイナリ構造であり、名称はJSON由来であり"バイナリ型JSON"の略語である。→wikipedia
- 先ほど WebBrokerで出力したJSONをBSONに変えます。
- uses System.JSON.BSON 追加

JSON出力をBSONに移行(2)

- コード

```
var
  ja: TJSONArray; jo: TJSONObject; bsonw:  TBsonWriter; JsonReader: TJsonStringReader;
Begin
  FJsonOut.AddPair('c1', ja);
  Response.ContentStream  := TBytesStream.Create;
  bsonw := TBsonWriter.Create(Response.ContentStream);
  JsonReader  := TJsonStringReader.Create(FJsonOut.ToString);
  bsonw.WriteToken(JsonReader);
  try
    Response.ContentType  := 'application/octet-stream';
    Response.ContentStream.Position := 0;
    Response.SendResponse;
  except
  end;
end;
```

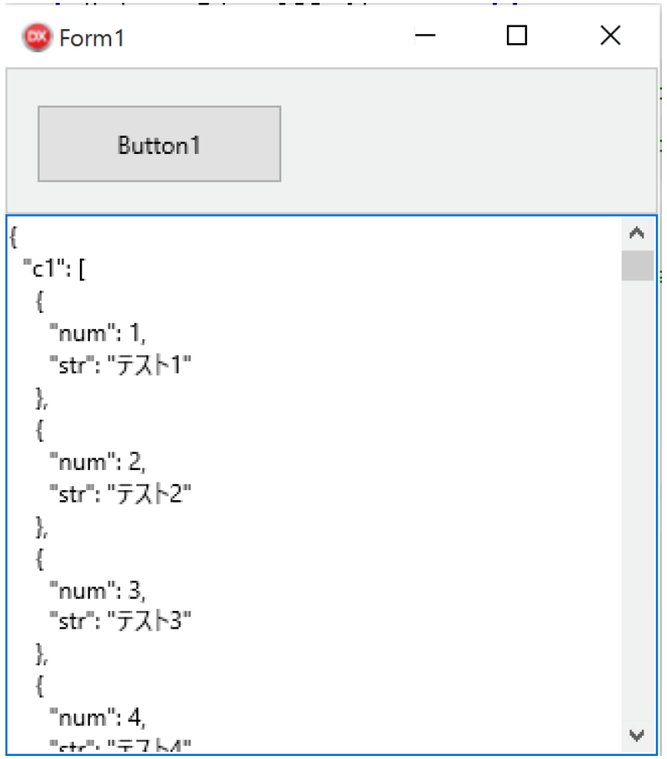
BSONをTNetHTTPClientで受信 (1)

- BSONをDelphiで受信するコードです

```
Begin
  bson  := TBytesStream.Create();
  NetHTTPClient1.Get('http://127.0.0.1:8080', bson);
  BsonReader := TBsonReader.Create(bson);
  JsonWriter := TJsonStringWriter.Create;
  JsonWriter.Formatting := TJsonFormatting.Indented;
  JsonWriter.WriteToken(BsonReader);
  Memo1.Lines.Text  := JsonWriter.ToString;
end;
```

BSONをTNetHTTPClientで受信 (2)

- 下絵の様にBSONを解釈してJSON表示できます



WebBroker + Fire Monkey の理由

- 多角形などのTPathが使える
- 透過PNGファイルが簡単に作れる
- グラデーションが使える
- ローテーションなどが楽に作れる

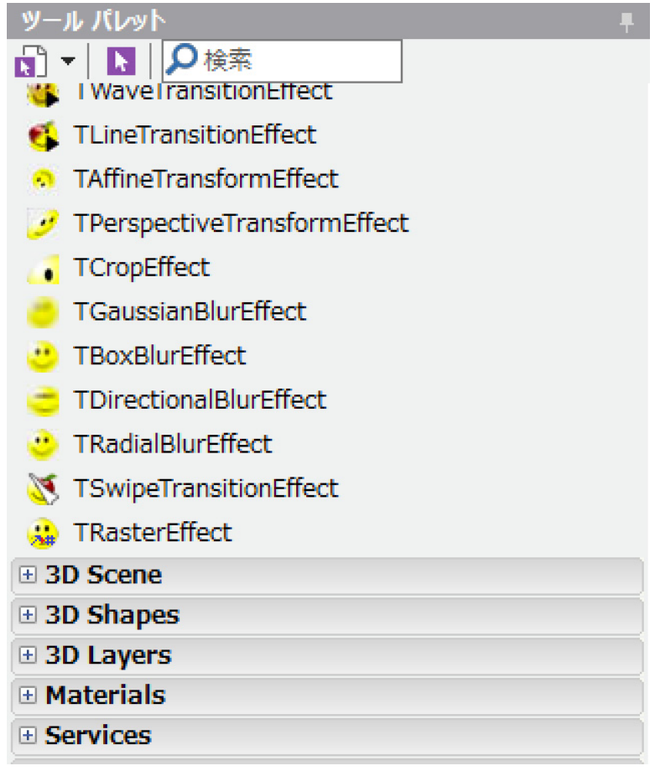
- Fire Monkey には沢山のメリットがありました。

WebBroker + FireMonkey(1)

- 5種類のアプリケーションが作れます
- 弊社の場合IISメインですので
  ISAP と スタンドアロンCGI実行 を使っています

WebBroker + FireMonkey(2)

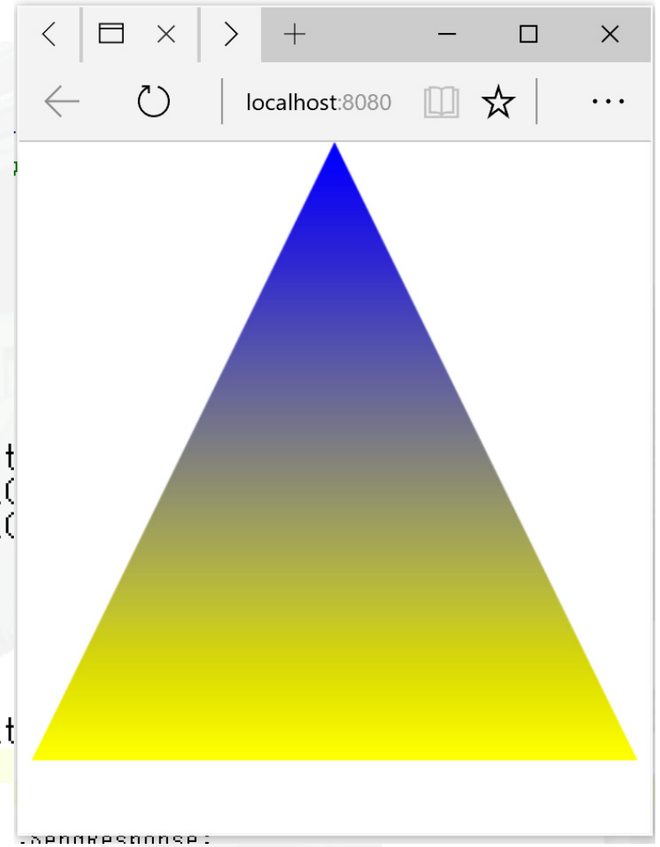
- プロジェクトを作成するとツールパレットにはFMX用のコンポーネントが出てきます。



三角形の透過PNGファイルを作る

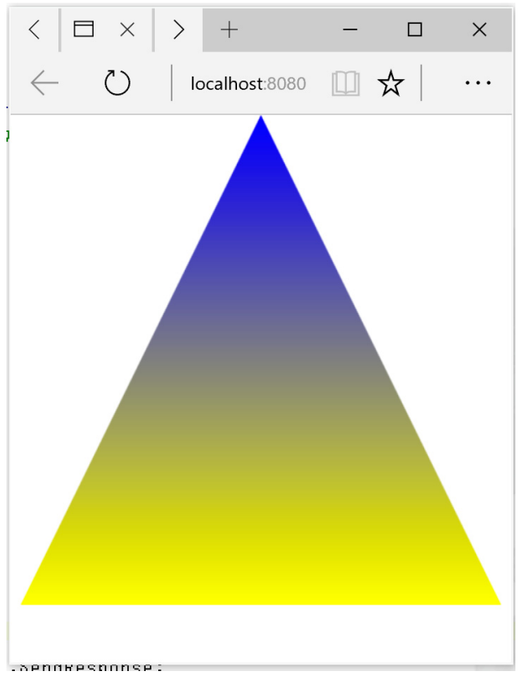
- これだけで三角のPNGファイルが完成です。

```
var
  pathdata: TPathData;
begin
  Response.ContentType  := 'image/png';
  pathdata  := TPathData.Create;
  try
    pathdata.Clear;
    pathdata.MoveTo(TPointF.Create(250,0));
    pathdata.LineTo(TPointF.Create(490,490));
    pathdata.LineTo(TPointF.Create(10,490));
    pathdata.ClosePath;
    FBmp.Canvas.BeginScene();
    FBmp.Canvas.Fill.Kind := TBrushKind.Gradient
    FBmp.Canvas.Fill.Gradient.Color   := TAlpha(
    FBmp.Canvas.Fill.Gradient.Color1  := TAlpha(
    FBmp.Canvas.FillPath(pathdata, 1);
    FBmp.Canvas.EndScene;
  finally
    pathdata.DisposeOf;
  end;
  Response.ContentStream  := TMemoryStream.Creat
  FBmp.SaveToStream(Response.ContentStream);
  Response.ContentStream.Position := 0;
  Response.SendResponse;
end;
```

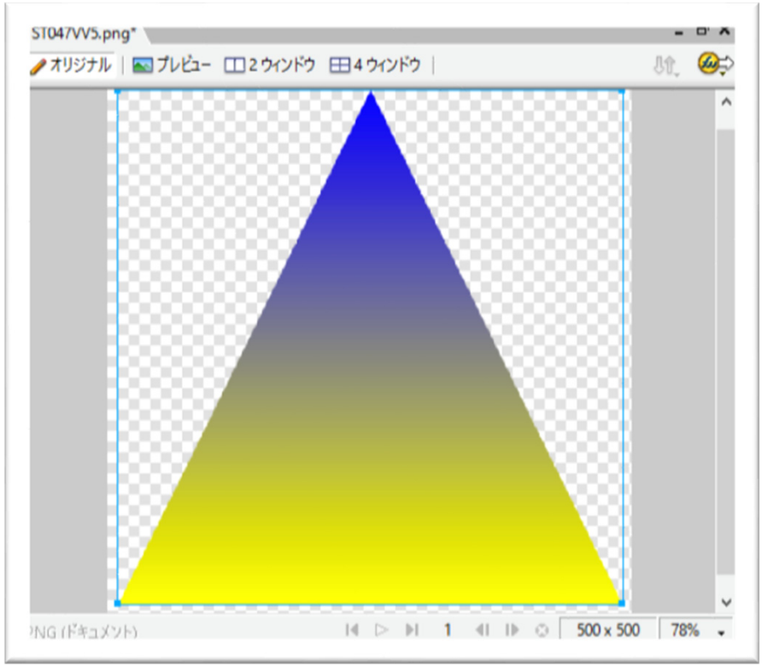


三角形の透過PNGファイル

- 透過になってるか他の画像編集ソフトで確認



ファイルを保存



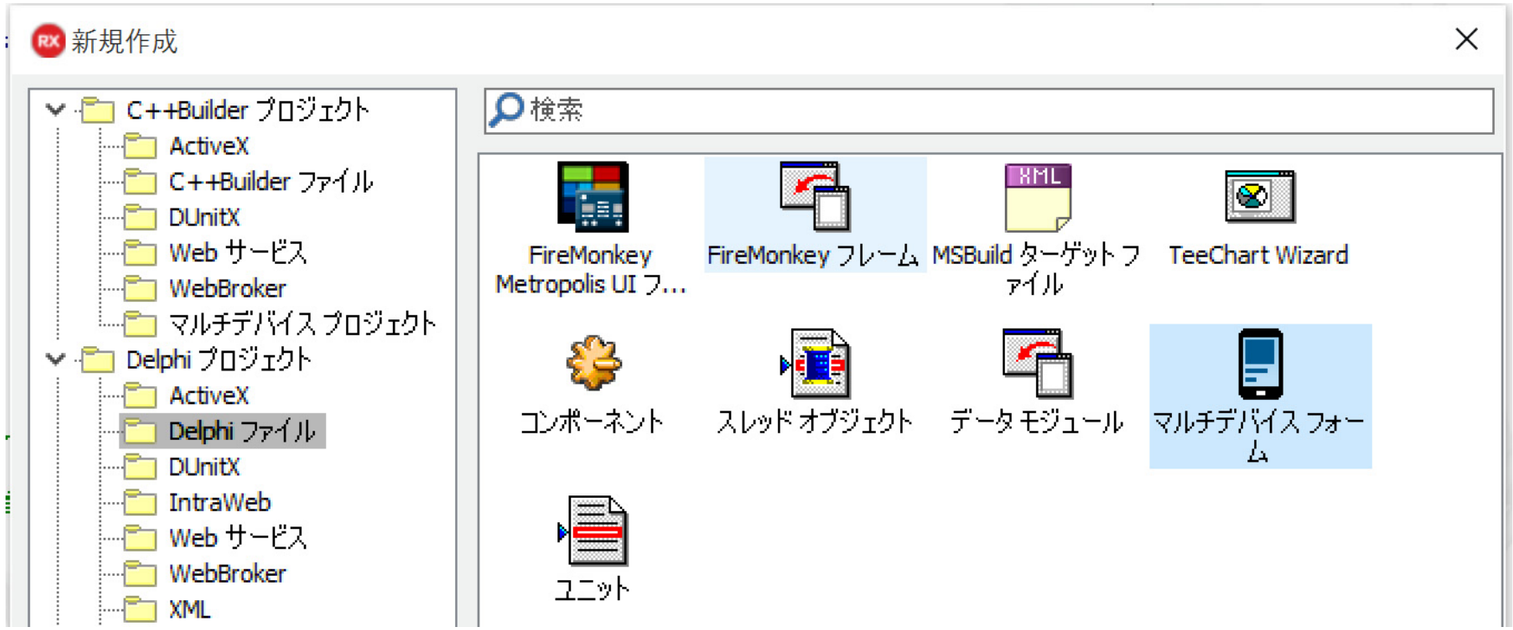
透過PNGで出力されている事が確認できました。

TBitmap.Canvasにガリガリ書けばいいのですが

- 至急の場合や先に確認が必要な場合が多々です。
- フォームに書いた内容がそっくりそのままをWeb出力すると
  IDEのフォームデザイナー側で設計できるので手間が省けます

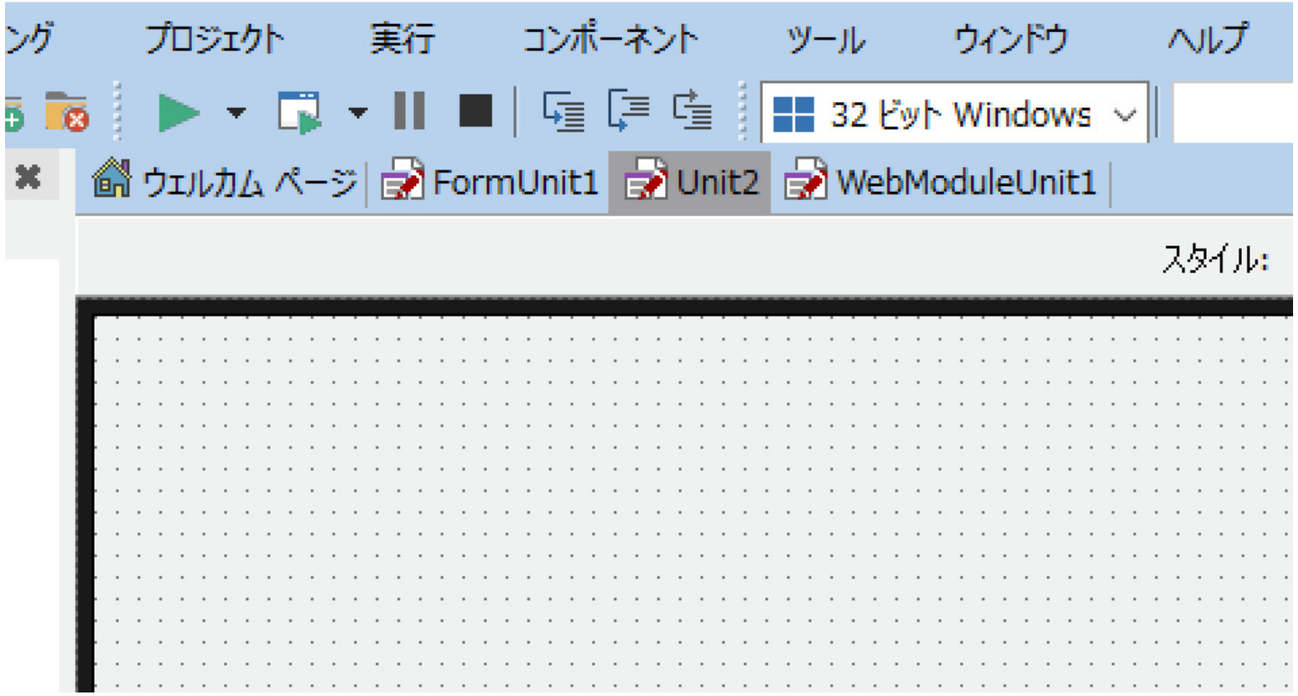
TWebModule以外にTForm追加したプロジェクト

- WebBrokerのプロジェクトに フォーム追加します



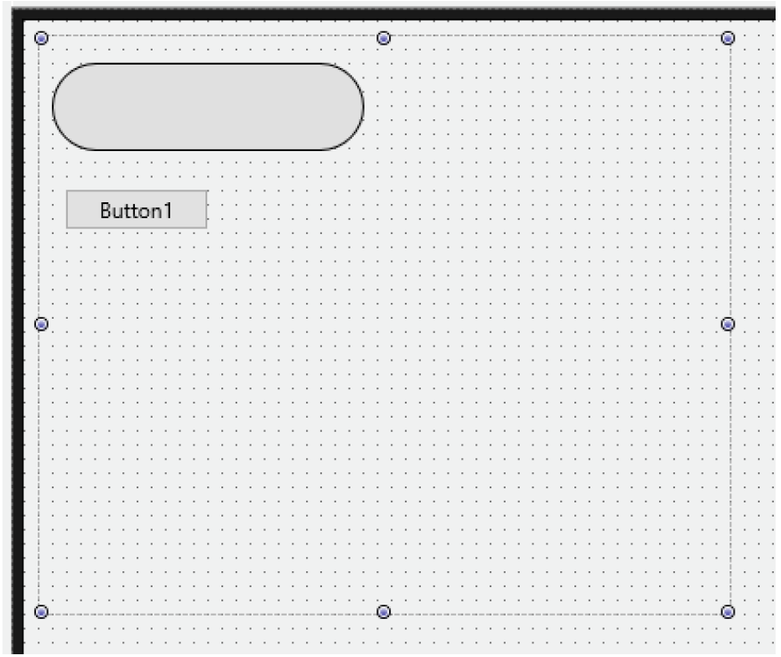
TWebModule以外にTForm追加したプロジェクト(2)

- WebModuleUnit1の隣にフォーム2が作られています



TWebModule以外にTForm追加したプロジェクト(3)

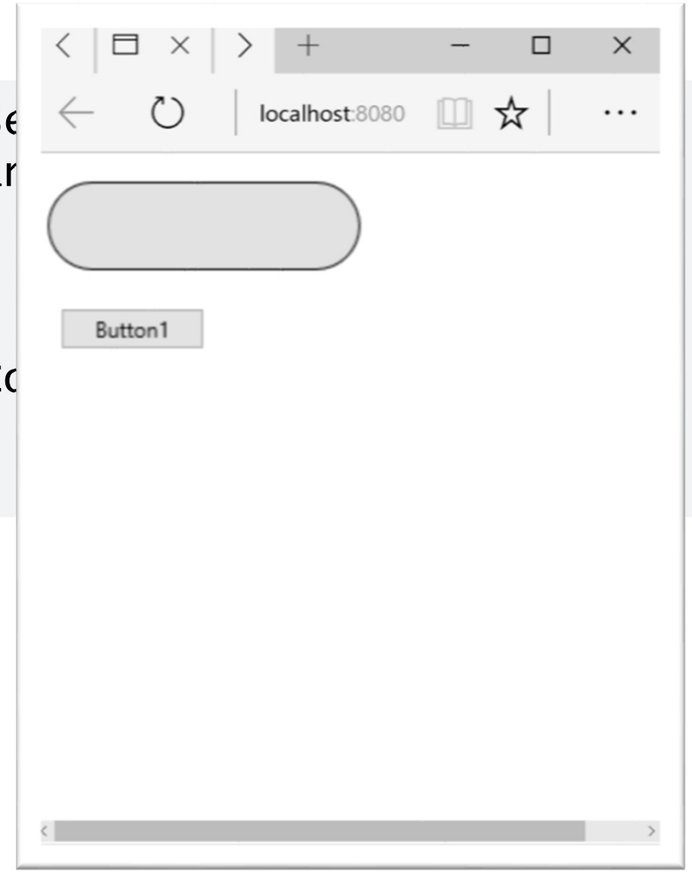
- フォーム2にTLayout, TRoundRect, TButtonを配置しました



TWebModule以外にTform追加したプロジェクト(4)

- 下記のコードを書いて実行してみます

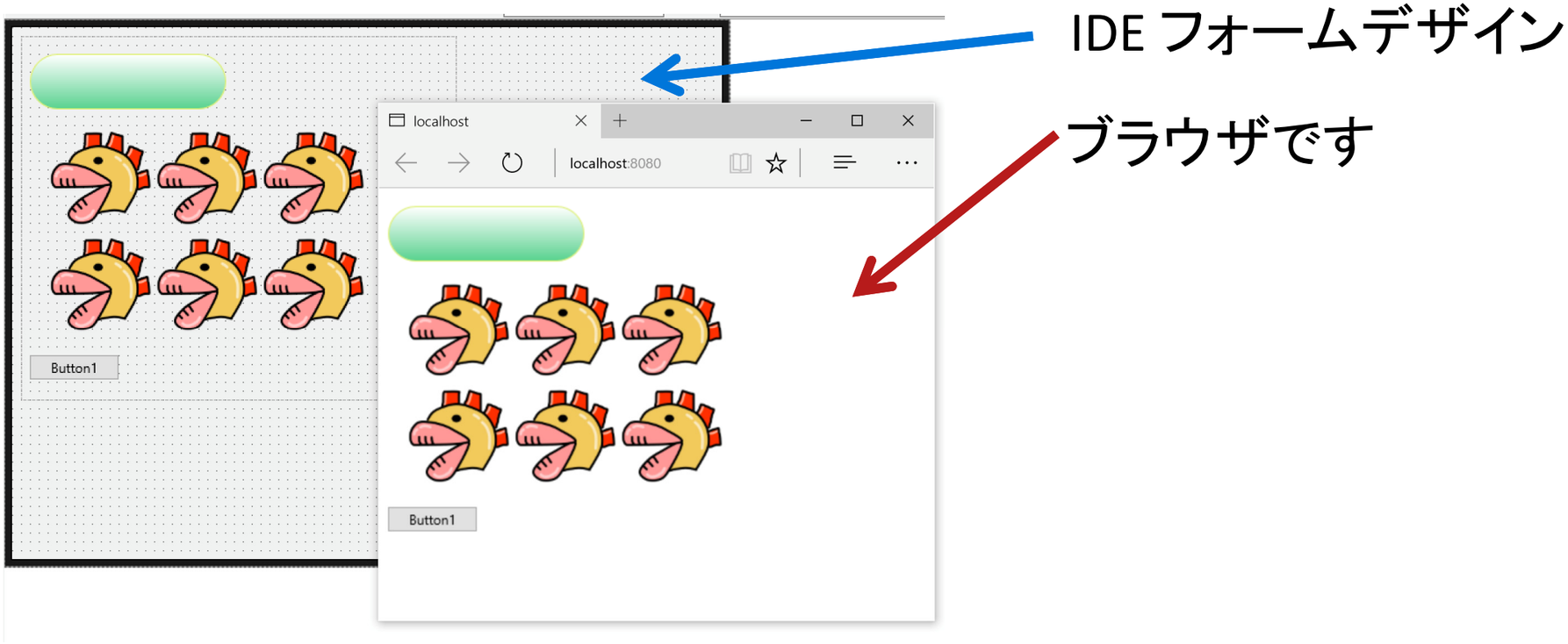
```
procedure TWebModule1.WebModule1DefaultHandlerAction(Se
  Request: TWebRequest; Response: TWebResponse; var Har
begin
  Response.ContentType    := 'image/png';
  Response.ContentStream  := TMemoryStream.Create;
  Form2.Layout1.MakeScreenshot.SaveToStream(Response.Co
  Response.SendResponse;
end;
```



デザイン側はIDEに任せています

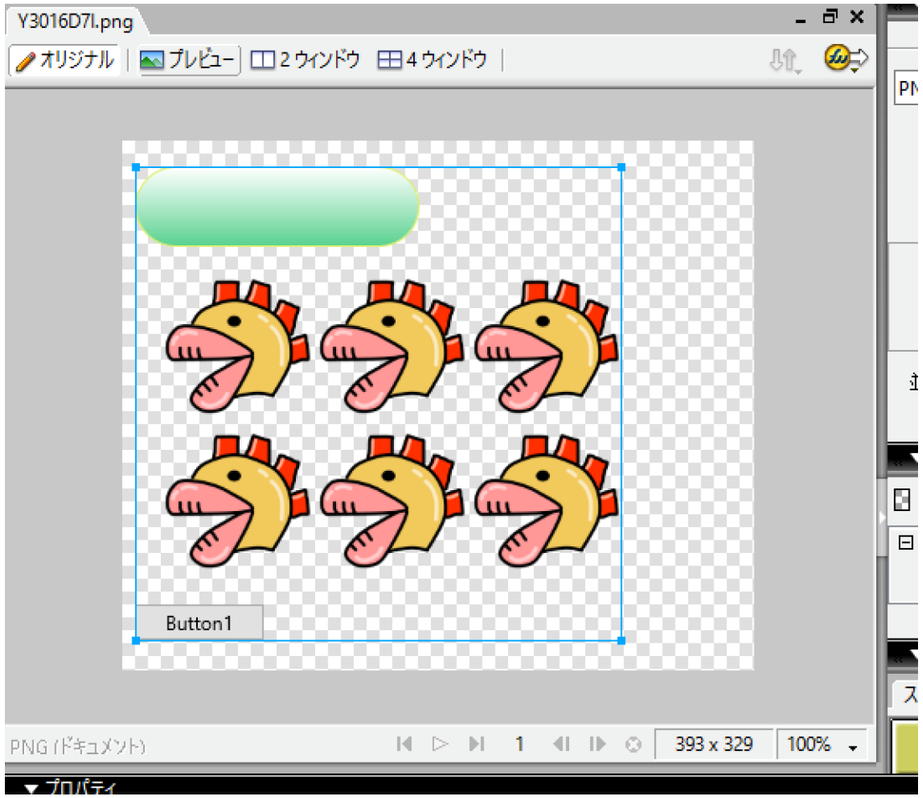
IDEのデザイン時と同じ

- IDEのデザイン時そのままなので作業時間短縮できます



透過の確認

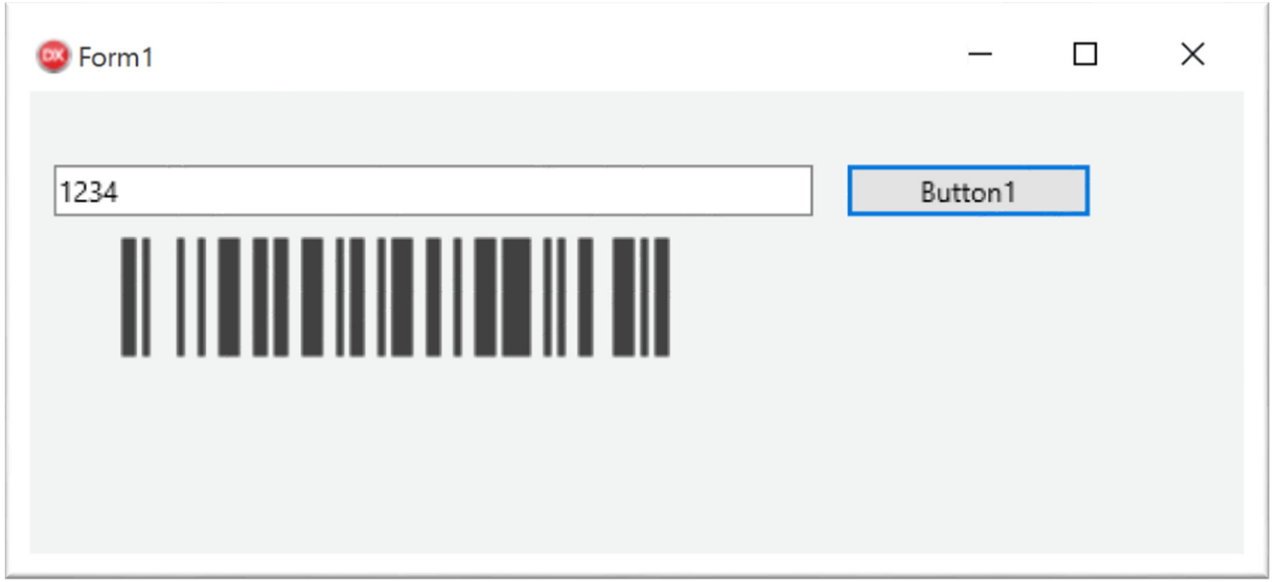
- ブラウザで出力されたデータを保存



他の画像ソフトで確認すると
きちんと透過されたPNGでした

バーコード表示

- DEKOさんのBARCODE.PASを使いました。



バーコードも同じくWeb側でPNG画像にする事が可能です。

バーコード表示(2)

- BARCODE.PASそのままのコードで動きます

```
  if not Assigned(Fbmp) then
    Fbmp  := TBitmap.Create(500, 50);
  bcode := Make_Code128(UpperCase(Request.QueryFields.Value
  position  := 0;

  if Length(bcode) > 0 then
  begin
    Fbmp.Canvas.BeginScene();

    for Counter:=1 to Length(bcode) do
    begin
      LineWidth := ((StrToInt(bcode[Counter]) div 2) + 1) *
      if (StrToInt(bcode[Counter]) mod 2) = 0 then // 空白
      begin
        Position := Position + LineWidth;
      end
      else
      begin //バー
        for i:=1 to LineWidth do
        begin
          Fbmp.Canvas.StrokeThickness  := 1;
          Fbmp.Canvas.StrokeDash := TStrokeDash.Solid;
          Fbmp.Canvas.DrawLine(TPointF.Create(Position + 30
          Inc(Position);
        end;
      end;
    end;
    Fbmp.Canvas.EndScene;
  end;
  Response.ContentType  := 'image/png';
```

バーコード表示(3)

- クーポン画像の上にバーコードをDrawBitmapで貼れば完成
- バーコード管理画面で指示したバーコードが反映されます

「「カラオケの鉄人」店舗で、クーポン作成―DTP並みの印刷システム実現の手法」

まとめ

まとめ

- 何故Webで実装したのか
  - むやみに実行ファイルを店舗に配布できない。
    - 店舗や本社も同じですが 配布するには時間もかかります
- フォームを使って画像イメージを作成するメリット/デメリット
  - 本社にてその場で確認できます
    - フタッフやデザイナーとIDE画面を返して確認しながら作業を進める事ができます。

31ST EMBARCADERO DEVELOPER CAMP

第 31 回　エンバカデロ・デベロッパーキャンプ