尾崎 浩司

株式会社ミガロ.

RAD事業部 営業·営業推進課

[Delphi/400] RESTによるWebサービスを活用 した機能拡張テクニック

1. はじめに 2. REST による Web サービスとは? 3. REST 機能を利用する方法 4. IBM Watson API 活用方法 5. REST 機能をもつコンポーネントの作成 6. さいごに



1973年8月16日生まれ 1996年3月三重大学工学部卒業 1999年10月株式会社ミガロ、入社 1999年10月システム事業部配属 2013年4月RAD事業部配属

現在の仕事内容: ミガロ.製品の素晴らしさをアピー ルするためのセミナーやイベントの 企画・運営等を主に担当している。

1.はじめに

近年アプリケーションの開発におい て、Web サービスの活用が盛んになっ ている。Web サービスとは、インター ネット技術を応用し、他のWeb サイト 上のソフトウェアを呼び出して利用する 仕組みのことである。【図 1】

現在では、大量のデータを蓄積してい る業者等が、そのデータを Web サービ スの形で一般のユーザーやプログラマー に提供する事例が多くなっている。たと えば、ネットショッピングサイト大手の アマゾンには「Product Advertising API」、楽天には「楽天市場商品検索 API」といった Web サービスがあり、 これら Web サービスを利用すると、サ イト上の商品検索等を自分のプログラム に組み込むことができる。さらに近年で は、従来の大量データをもつ業者だけで なく、IBM Watsonのような自然言語 を理解し、機械学習により人間の意思決 定を支援するシステムまでもが、Web サービスとして利用可能になっている。

Web サービスは、インターネット技 術を使用するのが特徴だが、その手法に はいくつかあり、代表的なのが SOAP (Simple Object Access Protocol) お よび REST (REpresentational State Transfer) である。

SOAP は、SOAP メッセージという XML によってメッセージ交換を行う方 法で事前にやり取りの定義が必要なた め、難易度が高い。最近は、よりシンプ ルな REST が主流である。本稿では、 Delphi/400 を使用した REST による Web サービスの使用方法や機能拡張方 法について説明する。なお、本稿のプロ グラムは Delphi/400 10 Seattle 以降の 環境を前提としている。

2.RESTによる Webサービスとは?

REST とは、Web サービスの設計モ デルのことで、ネットワーク上のデータ (リソース)を一意な URL で表すもの である。サービスの URL に HTTP メ ソッドでアクセスすることでデータの送 受信が行える。パラメータを指定して URL にアクセスすると特定の形式で データが返ってくるものだ。データ形式 には、XML あるいは JSON が利用可能 であるが、近年は、よりシンプルな JSON が使われることが多い。

JSON と は、JavaScript Object Notation の略で、軽量のデータ交換 フォーマットのことである。key (名前) と value (値) を「:」で対にして記述し、 まとまりごとに {) で囲うといった表記 法で、[] で配列を表現することもできる。 たとえば、【図 2】のような JSON は、 "result" というキーの配列の第一要素の 中にある "score" というキーの値が 80 であると解釈できる。

では、ここで REST + JSON による Web サービスを試してみる。livedoor が提供する「お天気 Web サービス」を 使ってみる。

http://weather.livedoor.com/weather_ hacks/webservice



(Google で "お天気 Web サービス" を検索すると上位に表示される「お天気 Web サービス仕様」。)

このサービスは、現在全国142カ所 の今日・明日・明後日の天気予報・予想 気温と都道府県の天気概況情報を提供す るものである。

Chrome ブラウザを立ち上げてアドレ ス欄に

[http://weather.livedoor.com/forecast/ webservice/json/v1?city=130010]

と入力してアクセスを行う。すると、【図 3】のような JSON が表示される。

これは、地域 ID=130010(東京都) の天気予報情報にアクセスした結果の JSON である。天気や最高気温などの情 報が JSON の中に含まれていることが わかる。このように REST + JSON に よる Web サービスは、URL にパラメー タを付けて呼び出すとレスポンスとして JSON データが返ってくることが確認で きる。

3.REST機能を 利用する方法

では、この「お天気 Web サービス」 を Delphi/400 から使用する方法を検討 する。Delphi/400には、RESTによる Web サービスを使用するためのコン ポーネントが用意されている。それが、 「TRESTClient」、「TRESTRequest」そ して「TRESTResponse」である。【図 4】 「TRESTClient」は、Web サービス へのリクエストを実行するコンポーネン トで、サービスに対する HTTP 接続を 管理し、HTTP ヘッダーおよびプロキ シ サーバーを処理し、応答データを受 け取るものである。「TRESTRequest」 は、HTTPリクエストを形成するパラ メータや設定をすべて保持する。 「TRESTResponse」は、Web サービス からのすべての戻りデータを保持する。 実際の設定は次のようになる。【図5】 ポイントは、RESTClient1 コンポー ネントの BaseURL プロパティに WEB サービスの基底 URL を指定すること、

RESTRequest1 コンポーネントの

Method プロパティに HTTP メソッド

の種類を、Resource プロパティに実行 パラメータを指定することである。

このプログラムの[検索]ボタンク リック時の処理は、【ソース1】となる。

1行目は、画面上で指定した地域 ID を Resource プロパティに記した "{CITY}"にセットする処理である。 Params プロパティの AddItem メソッ ドがリクエストのパラメータを定義する メソッドである。2行目は、Web サー ビスへのリクエスト実行になり、レスポ ンスの JSON 文字列を取得して Memol にセットするのが、3行目である。実際 に実行した結果が【図6】となる。

REST による Web サービスによって JSON データが取得できることを確認し たが、実際にはこの JSON データをパー ス (解釈) し必要な情報を抜き出す必要 がある。Delphi/400 にはこの JSON を 取り扱うためのユニットが用意されてい る。それが、"System.JSON"ユニットで ある。この中に、JSON オブジェクトを 実装したクラス TJSONObject や、文 字列、数値、オブジェクト、配列、 true/false の型を持つすべての JSON クラスの上位クラス TJSONValue が用 意されているので、これらを使用するこ とでパースすることが可能である。

たとえば、【図 7】が JSON をパース して特定のキーの値を取得するロジック 例である。

お天気情報の中から、今日と明日と明 後日の天気を取得して表示する処理を実 装してみる。

「お天気 Web サービス」の説明 Web ページを確認すると、レスポンスの仕様 が記載されている。[forecasts] プロパ ティの中にある [date] プロパティが予 報日、そして [telop] プロパティが天気 である。なお、3日分のデータは配列と して定義されている。この情報をもとに JSON をパースして、天気予報を画面に 表示するように改良したのが、【ソース 2】である。完成したプログラムを実行 すると、【図 8】のように指定した地域 の3日分の天気予測を取得することがで きる。

もう1つWebサービスの例を紹介す る。「HeartRails Geo API」(http:// geoapi.heartrails.com/)である。これは、 郵便番号/住所/緯度経度データ等の地 理情報を提供するWebサービスである。 この中に、「最寄駅情報取得 API」とい う機能があり、これは郵便番号を指定す ると、その地区の一番近い最寄駅がわか る機能である。これを活用すると、たと えば社内の取引先マスタにある郵便番号 を使用して、同じ最寄駅から歩いて訪問 できる取引先をピックアップする使い方 ができる。

サンプルプログラムを紹介する。 REST コンポーネントの設定は、【図 9】、 プログラムは【ソース 3】となる。この API は、パラメータ postal に郵便番号 をセットし呼び出すと、レスポンスとし て、response プロパティの station 配 列の中にある prefecture プロパティに は都道府県が、line プロパティに路線が、 そして name プロパティには最寄駅が セットされる。作成したプログラムを実 行して、郵便番号を入力し、[検索]ボ タンをクリックすると、最寄駅が表示さ れることがわかる。【図 10】

この最寄駅情報取得 API も、先ほど の天気情報の Web サービスと全く同じ やり方で処理ができることがわかる。こ のように、REST + JSON の Web サー ビスは、とても簡単に使用できるので、 いろいろなサービスを試してみていただ きたい。本稿執筆にあたり、Web サー ビスを調査したが、「API List 100+」 (http://smsurf.app-rox.com/api/) と いうサイトが役立った。ここにはいろい ろな Web サービスが一覧掲載されてい るので、便利な機能を見つけてみてほし い。【図 11】

4.IBM Watson API 活用方法

「コグニティブ」という言葉を聞いた ことがないだろうか。日本語では「認知」 のことで、ある事象についてコンピュー タが自ら考え、学習し、自らの答えを導 き出すシステムのことをいう。身近なと ころでは、iPhone の Siri やスマートス ピーカー等が有名である。従来システム との本質的な違いは、音声・画像・文章 等の非定型データも処理できることであ る。従来システムがもつ定型データと組 み合わせることで、人の作業を補助し、 より便利なシステム構築が可能になる。 この「コグニティブ」分野で IBM が提 供するのが、Watson である。この



Watson も Web サービスとして利用す ることができるので本章で説明する。

Watson は、IBM のクラウドサービ ス(IBM Cloud)上で API として提供 されており、クラウドサービスに登録す れば誰でも使用できる。本格的な利用に は有償プランが必要だが、IBM Cloud の各種サービスを無料で使用できる「ラ イト・アカウント」があるので、こちら を使用するとよい。「ライト・アカウント」 には、サービス使用量や機能の制約はあ るが、クレジットカード登録不要で簡単 に登録できる。

https://www.ibm.com/cloudcomputing/jp/ja/bluemix/lite-account/ (Google で"IBM Cloud ライト・ア カウント"を検索すると上位に表示され る「IBM Cloud ライト・アカウント -Japan」。)

2018 年 8 月時点で確認した Watson の API が【図 12】である。調べたすべ てのサービスが「ライト・アカウント」 で使用可能だ。

今回は、Language Translator サー ビスを例に説明する。これは言語変換(翻 訳)サービスで、特長としては、一般的 なWEB翻訳サービスと違い、専門用語 等を個別に登録することや、機械学習に よるカスタム翻訳モデルを作成すること が可能で、高精度の翻訳が行えることで ある。このサービスを活用すると、海外 担当者とのやり取り時の自動翻訳や、シ ステムの多言語対応で、DB上に日本語 でしか保持していない情報を翻訳して、 画面に出力することができる。

Watson は、サービスごとにインスタ ンスの作成が必要である。まず IBM Cloud (https://console.bluemix.net/) からサインインを行い、表示されたダッ シュボード画面より、[リソースの作成] ボタンをクリックする。Watsonをはじ めとするサービス選択画面が表示される ので、[Language Translator]を選択す る。サービス概要画面が表示され、プラ ンの選択ができるようになるため、「ラ イト」が選択されていることを確認し、 [作成]をクリックすれば、完了である。 インスタンスの作成が完了すると、作成 したサービスの管理画面が表示される。 【図 13】 【図 13】の管理画面の中にある「資格 情報」が重要である。2018年8月現在サー ビスによって「資格情報」には2種類あ り、ユーザーとパスワードが表示される サービスと、API Key が表示されるサー ビスとがある。Language Translator サービスの場合、API Key が表示され るため、この API Key と URLを控え ておけばよい。

また、管理画面には「APIリファレ ンス」画面へのリンクがあり、そこにア クセスすれば APIの仕様が記載されて い る。2018 年 8 月 現 在 Language Translator サービスは、V3 というバー ジョンになっており、APIの仕様は、【図 14】のとおりである。

Delphi/400から利用するポイントを 説明する。今回は、画面上に日本語で入 力したテキストを英語に翻訳するアプリ を作成する。(英語から日本語への翻訳 も可能にする。)

WatsonAPIの場合、資格情報が必要 なため、認証が必要である。 WatsonAPIでは、基本認証を使用する ことができ、これは「THTTPBasic Authenticator」コンポーネントが使用 できる。

REST コンポーネントの設定は【図 15】のとおりである。

サービスの資格情報がユーザーとパ ス ワ ー ド の 場 合 は、 そ の ま ま 「THTTPBasicAuthenticator」コンポー ネ ン ト の Username プ ロ パ テ ィ と Password プロパティにセットすればよ く、APIKey の 場 合 は、Username プ ロパティに "apikey"、Password プロパ ティに資格情報の APIKey を入力すれ ばよい。

今回のアプリの画面レイアウトは、 【図 16】、プログラムは【ソース 4】の ようになる。この Web サービスは、 POST メソッドとなり、リクエスト本 体に JSON 形式でパラメータを渡すと ころがポイントである。

実際に実行したアプリケーションが、 【図 17】である。Watson APIを利用す るアプリケーションも REST + JSON で簡単に構築できることがわかる。

5.REST機能をもつコ ンポーネントの作成

今回、REST + JSONを使用したWeb サービスの活用方法について具体例を挙 げながら説明したが、Web サービスの 課題点は、サービス提供者の都合により、 サービスの仕様が変更されたり、サービ ス自体が終了してしまう可能性があるこ とだ。

ある Web サービスを活用したアプリ ケーションを使用していた場合に、この ような事態が発生すると、新しい仕様に あわせてプログラムを変更したり、ある いは代替サービスに置換したりといった 作業が必要になる。こういったことを想 定した場合、Web サービスの機能を個々 のプログラムに都度記述する方法だと、 修正ボリュームが多大になることが想像 できる。

また、プログラムの中から Web サー ビスの部分を抜き出して修正しなければ いけないため、煩雑な作業になることが 予想される。

この問題を解決するには、どうすれば よいか? 1つの方法が各 Web サービ スごとにコンポーネント化してしまうこ とである。そうすれば、Web サービス の仕様変更時にも、コンポーネントソー スのみ修正し、各プログラムは、リコン パイルだけすれば済むはずである。

今回は、コンポーネント化の例とし て、前章で使用したLanguage Translatorサービスのコンポーネント 化を検討する。(TComponentを継承し たTTranslatorコンポーネント(非ビ ジュアルコンポーネント)を作成する。)

本稿では、コンポーネントそのものの 基本的な作成手順については割愛する が、作成手順が分からない場合は、2012 年度版『ミガロ.テクニカルレポート』 の SE 論文「カスタマイズコンポーネン ト入門」を参照していただきたい。

宣言部は、【ソース5】となる。変換 元の言語(SourceLanguage プロパティ) と変換後の言語(TargetLanguage プ ロパティ)、そして変換対象の文字列 (Source プロパティ)を設定した後、 Translate メソッドを実行すると翻訳が 行われ、その結果は、Destination プロ パティにセットされるという仕様を想定 している。

ソース1 検索ボタンのOnClickイベント uses REST.Types; procedure TForm1.Button1Click(Sender: TObject); begin //URLパラメータの指定 RESTRequest1.Params.AddItem('CITY', Edit1.Text, pkURLSEGMENT); //リクエスト実行 RESTRequest1.Execute; //レスポンスJSONを表示 Memol.Text := RESTResponsel.JSONText; end; 図6 お天気情報取得実行例 地域ID 130010 検索 "link":"http://weather.livedoor.com/area/forecast/130010", "forecasts": ["dateLabel":"今日", "telop":"晴のち曇", "date":"2018-08-25", "temperature": "min":null, "max": { < 図7 JSON パース例 変数sRet の値 procedure TForm1.Button1Click(Sender: TObject); 'result":[var JSONValue: TJSONValue: 'subject":"english", "score": 80, begin //*JSONデータのパース* JSONValue := TJSONObject.ParseJSONValue(sRet); *//結果の取得* "state":"good" 3, ShowMessage(JSONValue.GetValue<string>('result[0].subject')); Ł end: "subject":"math", "score": 70, "state":"normal" 実行結果 english } OK

実装部は、【ソース 6】および【ソー ス7】となる。コンポーネントの生成時 (Create メソッド)において、内部的に REST コンポーネントを生成し、 Language Translator サービスの仕様 に基づいたパラメータの設定を行ってい る。あとは、Translate メソッドにて、 【ソース 4】と同様の変換処理を行って いる。

このコンポーネントを使用したサン プルプログラムは、とてもシンプルであ る。画面レイアウトは、【図 18】、プロ グラムは【ソース 8】である。Web サー ビス自体をコンポーネント化しているた め、APIの仕様部分はこのプログラム には含まれていないことがわかる。これ によって、将来 Web サービスが終了し ても、コンポーネントの内容を別の Web サービスに変更すれば、個々のプ ログラムを変更する必要がなくなり、耐 性の強いプログラムであることがわかる だろう。

6.さいごに

本稿では、REST による Web サービ スを活用した Delphi/400 の機能拡張と して、いくつかの Web サービスを使用 した具体例を紹介してきた。REST + JSON 方式が簡単に Delphi/400 から活 用できることがわかる。単純に REST コンポーネントを組み込むだけでも十分 活用できるが、コンポーネント化まで検 討することにより、より耐性の強い仕組 みが作れるのである。ぜひ本稿を参考に いろいろな Web サービスの活用をご検 討いただきたい。

Μ

uses REST.Types, System.JSON;
procedure TForm1.Button1Click(Sender: TObject); var JSONValue: TJSONValue; begin //URLパラメータの指定 PESTRoquest1 Parama AddItom('CLIV' Edit1 Text pkURLSEGMENT);
//リクエスト実行 RESTRequest1.Execute; //レスポンスJSONを表示
//天気予報の取得 EditDate1.Text := JSONValue.GetValue <string>('forecasts[0].date'); EditDate2.Text := JSONValue.GetValue<string>('forecasts[1].date'); EditDate3.Text := JSONValue.GetValue<string>('forecasts[2].date'); EditWeather1.Text := JSONValue.GetValue<string>('forecasts[0].telop'); EditWeather2.Text := JSONValue.GetValue<string>('forecasts[1].telop'); EditWeather3.Text := JSONValue.GetValue<string>('forecasts[1].telop'); EditWeather3.Text := JSONValue.GetValue<string>('forecasts[2].telop');</string></string></string></string></string></string></string>

図8 お天気情報取得実行例2

🛞 Form1				×
地域ID 130010	検索			
今日 2018-08-25	明日 2018-08-26	明後日 2018-08-27]	
晴のち曇	晴時々曇	晴のち曇]	



10 最寄駅情報取得	
Form1	– 🗆 X
郵便番号	
1000013 検索	
都道府県 路線	最寄り駅
東京都東京火トロ日比谷線	霞ケ関
11 Webサービス 一覧サイト	
	0 - 0 ×
← → C ① 保護されていない通信 smsurf.app-rox.com/api/	☆ ▷ ♀ ♡ ▥ 匠 ♥ ♫ :
海外 · 国内 API一覧	G+ R市 M VANA1 92 メリイート B1731 ^
APILIST 100+	
公開されている気になるAPI/Webサービスをリスト おもしろそうなAPIを見つけたらつど追加して	ト化してみました いきますね
Developer Center	
Google Developers 提供: Google	ア AP1 キーワード検索 キーワードを1500
Android、iOS、Web環境での開発キットを提供。Google Play、Google+、	
Maps、You lube、Books、Gmail、Cloud などのAP1も多数Z3% Google Cloud 提供: Google	広告は Google により終了しま
コンピューティング、ストレージ・データベース、ネットワークといったインフ	この広告の表示を停止
ラから機械子当による方相よで様々な機能を提供 <u>Microsoft Azure</u> 提供: Microsoft	広告表示設定 ①
コンピューティング、ストレージ・データベース、ネットワークといったインフ	
ラルラ機械子書による方町まで様々な機能を提供 Bing for partners 提供: Microsoft	Page Top
地図、音声、翻訳、検索、Web管理、広告などのAPIを公開	
Amazon Developer 提供: Amazon	

Æ

 \oplus

I

-(

Ð

図12 Watson API一覧

分類	API種類	機能	ライト
照会応答系	Assistant (照会応答)	自然言語で対話可能なアプリケーションを、シンプルな開 発ツールで迅速に構築	対応
言語系	Language Translator (言語変換)	コンテンツのテキストを、ある言語から別の言語にリアル タイムで翻訳	対応
	Natural Language Understanding(自然言語理解)	テキスト分析を行い、コンテンツから概念、エンティ ティー、キーワード、カテゴリー、感情、関係、意味役割 などのメタデータを抽出	対応
心理系	Personality Insights (性格分析)	テキストから筆者のパーソナリティ(ビッグ・ファイブ、 価値、ニーズ)の3つの特徴を推測	対応
音声系	Speech to Text(音声認識)	ディープ・ラーニングを活用して、音声からテキストを書 き起こす	対応
	Text to Speech(音声合成)	テキストから自然な音声を合成	対応
知識探索系	Discovery(検索)	大量のデータを検索するとともに、適切な意思決定を支援	対応
	Knowledge Studio	業界や分野ごとの知識だけでなく、各分野の言葉の使われ 方の微妙な違いをWatsonに教えることができるツール	対応
画像系	Visual Recognition(画像認識)	ディープ・ラーニングを使用して、画像に写った物体・情 景・顔など様々なものを分析・認識	対応

※2018年8月現在

図13 Language Translatorサービス管理画面

IBM Cloud	カタログ 資料 サポート 管理	Q 尾崎	浩司's Account	\bigcirc
理 ービス資格情報 ラン	Language Translator / Watson / (学) Language Translator-wl リソース・グループ: Default 場所: 米国南部	3	:	•
H花	Get started with the service. チュートリアル <u>API リファレンス</u>	ブラン: Lite ア	ップグレード	
	資格情報		非表示 💋	
	API Key: saCYB7uQ	PQ769MxzH	ß	
	Url: https://gateway.watso	nplatform.net/language-transla	tor/ 🕒	
<				

POST /v3/translate		※2018年8 ※API仕様	3月現在 の詳細は、APIリ	ファレンスを参照	
パラメータ	タイプ		内容		
version	URLパラメ-	-タ	V3公開日(固定值:2018-	05-01)
request	リクエストの	の本体	翻訳に必要な	な下記パラメーク	タをJSON形式で指定
text	string []		翻訳したい	元のテキストを打	指定
source	string		元テキストの	の言語を指定 (j	a,en)
target	string		翻訳後の言語	語を指定 (ja,er	n)
レスポンス例		パラメ	タ	内容	
{ "translations" :[{ "translation" :"Hi	i,"	transla [transla word_u	tions: ation]	翻訳結果のう	テキスト D単語数
}],		charac	ter count	元テキストの	の文字数
} Language Translator לי גלטבטר אטגאספ TIPBasicAuthenticator1 THTTPBasicAuthenticator1	サービス REST	設定		パ オブジェクト インスペク RESTRequest1 TRESTR.	9 – D X rquest
} Language Translator ボオブシェクトインスペクタ TTPBasicAuthenticator1 THTTPBasicAuthenticator1 対策第 プロ(ディーイベント) BindSource HTTPBasicAuthenticator1 LiveBinding デザイナ LiveBinding デザイナ Name HTTPBasicAuthenticator1 Password saCYB7uQca Tag 0 Username aplikey	サービス REST ator	設定 HTTPBasicAuthentica	ator 1 RESTRequest 1	 パブジェクトインスペク RESTRequest1 TRESTR ゆ 検索 ブロバディ イベント Accept AcceptCharset AcceptEncoding AutoCreateParams B BindSource B Client 	9 ロ × equest ・ application/json, text/plain; q=0.9, text/1 ~ UTF-8, ";q=0.8 □ False RESTReques RESTReques POSTメンッド
} Language Translator ペ オブジェクト インスペクタ TTPBasicAuthenticator1 THTTPBasicAuthenticator1 D検索 プロバティ[イベント] BindSource HTTPBasicAuthenticator1 LiveBinding デザイナ Name HTTPBasicAuthenticator1 Password saCYB7uQca) Tag 0 Username apikey	サービス REST ator	設定 HTTPBasicAuthentica	ator 1 RESTRequest 1	 パーオブジェクト イソスペク RESTRequest1 TRESTR ク検索 ブロバティ イベント Accept AcceptCharset AcceptCharset AcceptCharset BindSource BindSource Client HandleRedirects EliveBinding デザイナ Method Name 	ターロン equest application/json, text/plain; q=0.9, text/1へ UTF-8, *;q=0.8 False RESTReques POSTメンッド RESTRequest I RESTRequest I
} Language Translator ペ オジェクトインスペタク TTPBasicAuthenticator1 THTTPBasicAuthenticator1 D接索 プロパティ イベント! BindSource LiveBinding デザイナ HTTPBasicAuthenticator1 LiveBinding デザイナ HTTPBasicAuthenticator1 Password Tag 0 Username apikey STCHent1 TRESTCHent LiveBinding	サービス REST	設定 HTTPBasicAuthentica	ator 1 RESTRequest 1 ACC RESTRESPONSE 1	 パーオブジェクト インスペク RESTRequest1 TRESTR ク検索 プロパティ イベント Accept AcceptCharset AcceptEncoding AutoCreateParams BindSource Client HandleRedirects LiveBinding デザイナ Method Name Params Resource Resource Resource 	ターロン equest application/json, text/plain; q=0.9, text/1 ← UTF-8, ";q=0.8 GRESTRequest RESTClient1 Interventioning デザイナ Interventioning Filter I
} Language Translator ペ オジェクトインスペクタ TTPBasicAuthenticator1 THTTPBasicAuthentic か後索 プロパティ イベント! BindSource HTTPBasicAuthenticator1 LiveBinding デザイナ HTTPBasicAuthenticator1 Password Tag 0 Username apikey STClient1 TRESTClient 地検索 プロパティ イベント! Accept	サービス REST	設定 HTTPBasicAuthentica	ator 1 RESTRequest 1 Liss RESTResponse 1	 パ オブジェクト インスペク RESTRequest1 TRESTR ク 検索 プロパティ イベント Accept AcceptCharset AcceptCharset AcceptEncoding AutoCreateParams BindSource BindSource Client HandleRedirects CliveBinding デザイナ Method > Name Params Resource Resource Resource Resource Resource SynchronizedEvents 	9 - ロ × equest application/json, text/plain; q=0.9, text/1 ~ UTF-8, ";q=0.8 False RESTRequest RESTClient1 I True LiveBinding デザイナ rmPOST RESTRequest1 (TRESTRequestParameterList) v3/translate?version=2018-05-01 RESTResponse1 False
	サービス REST	設定 HTTPBasicAuthentica	ator 1 RESTRequest 1 人 RESTResponse 1 実行パラメー	ペ オブジェクト インスペク RESTRequest1 TRESTR ク 検索 ブロバライ イベント! Accept AcceptCharset Accept AcceptEncoding AutoCreateParams B BindSource B Client HandleRedirects B LiveBinding デザイナ Method > Name Params Resource Resource SynchronizedEvents グ 検索	ターロン equest application/json, test/plain; q=0.9, test/1 ~ UTF-8, ";q=0.8 False RESTRequest RESTClient1 True LiveBinding デザイナ rmPOST RESTRequest1 (TRESTRequestParameterList) v3/translate?version=2018-05-01 RESTResponse1 False
	サービス REST	設定 HTTPBasicAuthentica	abor 1 RESTRequest 1 Liss RESTResponse 1 実行パラメー 基底URL	メオジェクト インスペク RESTRequest1 TRESTR ク 検索 プロパティ イペント! Accept AcceptCharset Accept AcceptEncoding AutoCreateParams B BindSource B Client HandleRedirects EureBinding デザイナ Method > Name Params Resource Resource Resource SynchronizedEvents メ オブジェクト インスペク RESTResponse1 TRESTF D 検索 プロパティ イベント! B BindSource ContentIngth ContentIngth ContentType B LiveBinding デザイナ	Point Stresponse 1.BindSource RESTResponse



Watsonを使用した翻訳アプリ	X	
ja ~		
今日の天気は曇り時々晴れる	です。明日は快晴になるでしょう。	
en v	· · · · · · · · · · · · · · · · · · ·	
Today's weather is cloudy	y and sometimes sunny. It will be fine	
tomorrow.		
	× .	
-ス5 TTranslatorコンポー	-ネントの宣言部	
-ス5 TTranslatorコンポー	ーネントの宣言部	
-ス5 TTranslatorコンポー unit Translator; interface	ーネントの宣言部 REST関連のユニット、JSONユニット	
-ス5 TTranslatorコンポー unit Translator; interface uses System.SysUtils, Syst	ーネントの宣言部 REST関連のユニット、JSONユニット	
-ス5 TTranslatorコンポー unit Translator; interface uses System.SysUtils, Syster REST.Authenticator.Bas	ーネントの宣言部 REST関連のユニット、JSONユニット tem.Classes, <u>System.JSON, RESI.Client.</u> asic, Rest.Types, IPPeerClient;	
-ス5 TTranslatorコンポー unit Translator; interface uses System.SysUtils, Syster <u>REST.Authenticator.Bas</u> type TLanguage = (Japanese	ーネントの宣言部 REST関連のユニット、JSONユニット tem.Classes, <u>System.JSON, REST.Client</u> , asic, Rest.Types, IPPeerClient; a, English): TComponentを継承 (まビジュアルコンポーネント)	
-ス5 TTranslatorコンポー unit Translator; interface uses System.SysUtils, Syste REST.Authenticator.Bas type TLanguage = (Japanese TTranslator = class(II private	ーネントの宣言部 REST関連のユニット、JSONユニット tem.Classes, <u>System.JSON, REST.Client,</u> asic, Rest.Types, IPPeerClient; a, English); IComponent) TComponentを継承 (非ビジュアルコンポーネント)	
-ス5 TTranslatorコンポー unit Translator; interface uses System.SysUtils, Syste REST.Authenticator.Bas type TLanguage = (Japanese TTranslator = class(II private { Private 宣言 } FRestClienct: TRESTI	-ネントの宣言部 REST関連のユニット、JSONユニット tem.Classes, <u>System.JSON, REST.Client</u> , asic, Rest.Types, IPPeerClient; e, English); TComponentを継承 (非ビジュアルコンポーネント)	
-ス5 TTranslatorコンポー unit Translator; interface uses System.SysUtils, Syste REST.Authenticator.Bas type TLanguage = (Japanese TTranslator = class(II private { Private 宣言 } FRestRequest: TRESTI FRestReponse: TREST FBasicAuthenticator	-ネントの宣言部 REST関連のユニット、JSONユニット tem.Classes, <u>System.JSON, REST.Client.</u> asic, Rest.Types, IPPeerClient; a, English): TComponentを継承 (非ビジュアルコンポーネント) IClient; Request; STResponse; : IHITPBasicAuthenticater: Translateメソッド:変換実行処理 Destinationプロパティ:変換後テキ	₽ - スト
-ス5 TTranslatorコンポー unit Translator; interface uses System.SysUtils, Syste REST.Authenticator.Bas type TLanguage = (Japanese TTranslator = class(II private { Private 宣言 } FRestClienct: TRESTI FRestResponse: TREST FBasicAuthenticator FSourceLanguage: TLi FTargetLanguage: TLi	-ネントの宣言部 REST関連のユニット、JSONユニット tem.Classes. <u>System.JSON, RESI.Client.</u> asic. Rest.Types. TPPeerClient: asic. Rest.Types, TComponentを継承 (非ビジュアルコンポーネント) IClient; Request; ITranslateメソッド:変換実行処理 Destinationプロパティ:変換後テキ	≝ -スト
-ス5 TTranslatorコンポー unit Translator; interface uses System.SysUtils, System REST.Authenticator.Base type TLanguage = (Japanese TTranslator = class(工 private { Private 宣言 } FRestClenct: TRESTI FRestRequest: TRESTI FRestResponse: TREST FBasicAuthenticator FSourceLanguage: TLi FSource: String; FDestination: String; function Catlanguage	-ネントの宣言部 REST関連のユニット、JSONユニット tem.Classes, <u>System.JSON, REST.Client</u> , asic, Rest.Types, IPPeerClient; a, English); IComponent) IClient; Request; STResponse; :: THTIPBasicAuthenticator; .anguage; anguage; Stresponse; :: THTIPBasicAuthenticator; .anguage; Stresponse; :: Thirpedicate (Alagonation); Streing;	₽ -スト
-ス5 TTranslatorコンポー unit Translator; interface uses System.SysUtils, Syste REST.Authenticator.Bai type TLanguage = (Japanese TTranslator = class(II private (Private 宣言) FRestClienct: TRESTI FRestResponse: TREST FRestResponse: TREST FBasicAuthenticator FSourceLanguage: TLa FTargetLanguage: TLa FSource: String; FDestination: Str	-ネントの宣言部 REST関連のユニット、JSONユニット tem.Classes, System.JSON, RESI.Client, asic, Rest.Types, TPPeerClient; a, English); IComponent IClient; Request; ITComponentを継承 (非ビジュアルコンポーネント) IClient; Request; ITranslateメソッド:変換実行処理 Destinationプロパティ:変換後テキ anguage; anguage; Sename(ALanguage: TLanguage): String; アロパティ Sourcel anguage: 変換元言	型 -スト
-ス5 TTranslatorコンポー unit Translator; interface uses System.SysUtils, Syste <u>REST.Authenticator.Ba</u> type TLanguage = (Japanese TTranslator = class(II private <i>{Private 宣言 }</i> FRestClienct: TRESTI FRestRequest: TRESTI FRestRequest: TRESTI FRestReponse: TREST FBasicAuthenticator FSourceLanguage: TLi FIargetLanguage: TLi FSource: String; FDestination: String function GetLanguage protected <i>{Protected 宣言 }</i> public <i>{Public 宣言 }</i>	-ネントの宣言部 REST関連のユニット、JSONユニット tem.Classes. <u>System.JSON, REST.Client.</u> asic. Rest.Types. <u>IPPeerClient:</u> asic. Rest.Types. <u>IPPeerClient:</u> (非ビジュアルコンポーネント) Client: Request: Stresponse: Translateメソッド:変換実行処理 Destinationプロパティ:変換後テキ SourceLanguage: 変換 定言 TargetLanguage: 変換 定言	里 ·スト i語
-ス5 TTranslatorコンポー unit Translator; interface uses System.SysUtils, Syste REST.Authenticator.Bas type TLanguage = (Japanese TTranslator = class(II private <i>Private</i> 宣言 / FRestClienct: IRESI FRestRequest: TRESI FRestReponse: TRESI FRasteLanguage: TLa FargetLanguage: TLa FSourceLanguage: TLa FSource: String; FDestination: String function GetLanguage protected <i>(Protected 宣言)</i> public <i>(Public 宣言)</i> constructor Create(<i>p</i> procedure Translate	-ネントの宣言部 REST関連のユニット、JSONユニット tem.Classes, System.JSON, RESI.Client, asic, Rest.Types, TPPeerClient: asic, Rest.Types, TPPeerClient: asic, Rest.Types, TPPeerClient: asic, Rest.Types, TPPeerClient: TComponentを継承 (非ビジュアルコンポーネント) IClient; Request; ITranslateメソッド:変換実行処理 Destinationプロパティ:変換後テキ anguage; asi seName(ALanguage: TLanguage): String: (AOwner: TComponent); override; String read Effectination:	型 ·スト i語
-ス5 TTranslatorコンポー unit Translator; interface uses System.SysUtils, Syste REST.Authenticator.Bas type TLanguage = (Japanese TTranslator = class(II private <i>f Private 宣言)</i> FRestClienct: TRESTI FRestRequest: TRESTI FRestRequest: TRESTI FRestRequest: TRESTI FRestResponse: TRES FBasicAuthenticator FSourceLanguage: TLi FTargetLanguage: TLi FTargetLanguage: TLi FTargetLanguage: TLi FDestination: String function GetLanguage protected <i>f Protected 宣言)</i> public <i>Constructor Create(i</i> procedure Translate property Destination published <i>f Published 宣言)</i>	-ネントの宣言部 REST関連のユニット、JSONユニット tem.Classes, <u>System.JSON, RESI.Client.</u> asic, Rest.Types, IPPeerClient: asic, Rest.Types, IPPeerClient: TComponentを継承 (非ビジュアルコンポーネント) IClient; Request; ITranslateメソッド:変換実行処理 Destinationプロパティ:変換後テキ SourceLanguage: 変換元言 TargetLanguage: 変換元言 Source: 変換元テキスト	型 - スト i語 語
-ス5 TTranslatorコンポー unit Translator; interface uses System.SysUtils, Syste REST.Authenticator.Bas type TLanguage = (Japanese TTranslator = class(王) private { Private 宣言 } FRestClienct: TRESTI FRestRequest: TRESTI FRestRequest: TRESTI FRestResponse: TRES FBasicAuthenticator FSourceLanguage: TLI FIargetLanguage: TLI FIargetLanguage: TLI Floestination: String; fDestination: String; function GetLanguage protected { Protected 宣言 } constructor Create(i procedure Translate property Destination published { Published 宣言 } property SourceLanguage	-ネントの宣言部 REST関連のユニット、JSONユニット tem.Classes. <u>System.JSON. REST.Client.</u> asic. Rest.Types. TPPeerClient: asic. Rest.Types. TPPeerClient: asic. Rest.Types. TPPeerClient: asic. Rest.Types. TPPeerClient: TCOmponentを継承 (非ビジュアルコンポーネント) TClient: Request: ITranslateメソッド:変換実行処理 Destinationプロパティ:変換後テキ anguage: SeName(ALanguage: TLanguage): String: (AOwner: TComponent); override: anguage: TLanguage read FSourceLanguage write FSourceLanguage; sugge: TLanguage read FSourceLanguage write FTargetLanguage;	型 スト 語 語



72

Yorditator: Yorditator:	DX Form1				- • ×	
Terestert Terestert Terestert Terestert 「WRN Terestert 「WRN Terestert Ware Terestert Start Terestert Terestert Teres						
1 1 1 1 <td< td=""><td></td><td>&</td><td></td><td></td><td>♪☆ オフジェクトインス — L</td><td></td></td<>		&			♪☆ オフジェクトインス — L	
Porter Translator1.Translator1.Destination; end;	· · · · ·	ranslator 1		I	ranslator1 I ranslator	· · · · · · · · · · · · · · · · · · ·
B 変換 EtreBinding 75/1* LiveBinding 75/1* ※ Name Tanslatori SourceLinguage Appanee Tanslatori Tanslatori SourceLinguage Appanee Tanslatori Tanslatori SourceLinguage Appanee Tanslatori Tanslatori SourceLinguage Expendition YoyNitta Tanslatori SourceLinguage Expendition YoyNitta Tanslatori SourceLinguage Expendition YoyNitta Tanslatori YoyNitta Ta				<u>د</u>	プロパティ イベント	
Name Translatoril 変換 Paparese Tagtinguage Inglish 747編集 747編集 747mm 747mm 747mm </td <td></td> <td></td> <td></td> <td>L L</td> <td>LiveBinding デザイナ LiveBinding デサ</td> <td>f゚゚゙イナ</td>				L L	LiveBinding デザイナ LiveBinding デサ	f゚゚゙イナ
State Surce Japanese Image: Image: Image:				>>	Name Translator1	
変換 Tag Tag 1/10/14集 Fr(CO)月目が表示されています 1/10/14集 Fr(CO)月日が表示されています 8 変換ボタンのOnClickイベント Procedure TForm1.ButtonTransrateClick(Sender: TObject); 1/2 変換スタンテ列のセット Translator1.Source := MemoSource.Text; 1/2 変換案行 Translator1.Translate; 1/1結果の表示 MemoTarget.Text := Translator1.Destination; end;					Source	
変換 「argetLanguage Inglish Provides Provides # # 変換ボタンのOnClick1ペント # Procedure IForm1.ButtonTransrateClick(Sender: TObject); begin //変換対象文字列のセット Translator1.Source := MemoSource.Text; //変換実行 Translator1.Translate; //結果の表示 MemoTarget.Text := Translator1.Destination; end;					Tag 0	
Yyy編 Yryy編 Yryy編 Yryyama Yryymama Yryyma Yryyna Yryyna Yry			715 +44		TargetLanguage English	
8 変換ボタンのOnClickイベント procedure TForm1.ButtonTransrateClick(Sender: TObject); begin //変換対象文字列のセット Translator1.Source := MemoSource.Text; //変換実行 Translator1.Translate; //結果の表示 MemoTarget.Text := Translator1.Destination; end;			2	n	小小年	
オペロ項目が表示されています オペロ項目が表示されています 8 変換ボタンのOnClickイベント procedure TForm1.ButtonTransrateClick(Sender: TObject); begin //変換対象文字列のセット Translator1.Source := MemoSource.Text; //変換実行 Translator1.Translate; //結果の表示 MemoTarget.Text := Translator1.Destination; end;					177桶未…	-
8 変換ボタンのOnClickイベント procedure TForm1.ButtonTransrateClick(Sender: TObject); begin //変換対象文字列のセット Translator1.Source := MemoSource.Text; //変換実行 Translator1.Translate; //結果の表示 MemoTarget.Text := Translator1.Destination; end;				ਰ	「べての項目が表示されています	
8 変換ボタンのOnClickイベント procedure TForm1.ButtonTransrateClick(Sender: TObject); begin //変換対象文字列のセット Translator1.Source := MemoSource.Text; //変換定行 Translator1.Translate; //結果の表示 MemoTarget.Text := Translator1.Destination; end;						
8 変換ボタンのOnClickイベント procedure TForm1.ButtonTransrateClick(Sender: TObject); begin //変換対象文字列のセット Translator1.Source := MemoSource.Text; //変換実行 Translator1.Translate; //結果の表示 MemoTarget.Text := Translator1.Destination; end;						
8 変換ボタンのOnClickイベント procedure TForm1.ButtonTransrateClick(Sender: TObject); begin //変換対象文字列のセット Translator1.Source := MemoSource.Text; //変換実行 Translator1.Translate; //結果の表示 MemoTarget.Text := Translator1.Destination; end;						
8 変換ボタンのOnClick1ベント procedure TForm1.ButtonTransrateClick(Sender: TObject); begin //変換対象文字列のセット Translator1.Source := MemoSource.Text; //変換実行 Translator1.Translate; //結果の表示 MemoTarget.Text := Translator1.Destination; end;						
8 変換ボタンのOnClick1ベント procedure TForm1.ButtonTransrateClick(Sender: TObject); begin //変換対象文字列のセット Translator1.Source := MemoSource.Text; //変換実行 Translator1.Translate; //結果の表示 MemoTarget.Text := Translator1.Destination; end;	::				· · · · · · · · · · · · · · · · · · ·	
8 変換ボタンのOnClickイベント procedure TForm1.ButtonTransrateClick(Sender: TObject); begin //変換対象文字列のセット Translator1.Source := MemoSource.Text; //変換実行 Translator1.Translate; //結果の表示 MemoTarget.Text := Translator1.Destination; end;						
8 変換ボタンのOnClickイベント procedure TForm1.ButtonTransrateClick(Sender: TObject); begin //変換対象文字列のセット Translator1.Source := MemoSource.Text; //変換実行 Translator1.Translate; //結果の表示 MemoTarget.Text := Translator1.Destination; end;						
8 変換ボタンのOnClickイベント procedure TForm1.ButtonTransrateClick(Sender: TObject); begin //変換文字列のセット Translator1.Source := MemoSource.Text; //変換実行 Translator1.Translate; //結果の表示 MemoTarget.Text := Translator1.Destination; end;						
procedure TForm1.ButtonTransrateClick(Sender: TObject); begin //変換対象文字列のセット Translator1.Source := MemoSource.Text; //変換実行 Translator1.Translate; //結果の表示 MemoTarget.Text := Translator1.Destination; end;	.8 変換ボタ	シのOnClickイベン	小			-
procedure Frormi.ButtoniransrateClick(Sender: TUbject); begin //変換対象文字列のセット Translator1.Source := MemoSource.Text; //変換実行 Translator1.Translate; //結果の表示 MemoTarget.Text := Translator1.Destination; end;		TC 1	D. L.L. T.		(0	
//変換対象文字列のセット Translatorl.Source := MemoSource.Text; //変換実行 Translatorl.Translate; //結果の表示 MemoTarget.Text := Translatorl.Destination; end;	nracea	ure irormi.	.buttoniransr	atellick	(Sender: IUbject	.);
Translator1.Source := MemoSource.Text; //変換実行 Translator1.Translate; //結果の表示 MemoTarget.Text := Translator1.Destination; end;	hegin					
//変換実行 Translator1.Translate; //結果の表示 MemoTarget.Text := Translator1.Destination; end;	begin //変	模对象文字。	列のセット			-
//変換美行 Translator1.Translate; //結果の表示 MemoTarget.Text := Translator1.Destination; end;	begin //変 Trans	<i>換対象文字。</i> slator1.Sou	<i>列のセット</i> urce := MemoS	Source.Te	×t;	
//結果の表示 MemoTarget.Text := Translator1.Destination; end;	begin //変 Trans	<i>換対象文字</i> slator1.Sou	<i>列のセット</i> urce := MemoS	Gource.Te	×t;	-
//結果の表示 MemoTarget.Text := Translator1.Destination; end;	begin //変 Trans	<i>換対象文字</i> slator1.Sou <i>換実行</i>	列のセット urce := MemoS	Source.Te	xt;	-
MemoTarget.Text := Translator1.Destination; end;	begin //変 Trans //変 Trans	<i>換対象文字。</i> slator1.Sou <i>換実行</i> slator1.Tra	<i>列のセット</i> urce := MemoS anslate;	Source.Te	xt;	-
end;	begin //変 Trans //変 Trans	<i>換対象文字</i> slator1.Sou <i>換実行</i> slator1.Tra <i>果の表示</i>	列のセット urce := MemoS anslate;	Gource.Te	xt;	-
	begin //変 Trans //変 Trans //結 Memo	<i>換対象文字</i> slator1.Sou <i>換実行</i> slator1.Tra <i>果の表示</i> [arget.Text	列のセット urce := MemoS anslate; t := Translat	Source.Te	xt; ination;	
	begin //変 Trans //変 Trans //結 Memo end;	<i>換対象文字</i> slator1.Sou <i>換実行</i> slator1.Tra <i>果の表示</i> Target.Text	列のセット urce := MemoS anslate; t := Translat	Source.Te	xt; ination;	
	begin //変 Trans //変 Trans //結 Memo end;	<i>換対象文字</i> slator1.Sou <i>換実行</i> slator1.Tra <i>果の表示</i> Target.Text	列のセット urce := MemoS anslate; t := Translat	Source.Te	xt; ination;	
	begin //変 Trans //変 Trans //結 Memo end;	換対象文字表 slator1.Sou <i>換実行</i> slator1.Tra <i>果の表示</i> Target.Text	列のセット urce := MemoS anslate; t := Translat	Source.Te	xt; ination;	
	begin //変 Trans //変 Trans //結 Memo end;	<i>換対象文字</i> slator1.Sou <i>換実行</i> slator1.Tra <i>果の表示</i> Target.Text	列のセット urce := MemoS anslate; t := Translat	Source.Te	xt; ination;	
	begin //変 Trans //変 Trans //結 Memo end;	換対象文字表 slator1.Sou 換実行 slator1.Tra <i>果の表示</i> Target.Text	列のセット urce := MemoS anslate; t := Translat	Source.Te	xt; ination;	
	begin //変 Trans //変 Trans //結 Memo end;	換対象文字表 slator1.Sou <i>換実行</i> slator1.Tra <i>果の表示</i> Target.Text	列のセット urce := MemoS anslate; t := Translat	Source.Te	xt; ination;	
	begin //変 Trans //変 Trans //結 Memo end;	<i>換対象文字</i> slator1.Sou <i>換実行</i> slator1.Tra <i>果の表示</i> Target.Text	列のセット urce := MemoS anslate; t := Translat	Source.Te	xt; ination;	
	begin //変 Trans //変 Trans //結 Memo end;	換対象文字表 slator1.Sou 換実行 slator1.Tra <i>果の表示</i> Target.Text	列のセット urce := MemoS anslate; t := Translat	Source.Te	xt; ination;	
	begin //変 Trans //変 Trans //結 Memo end;	<i>換対象文字</i> slator1.Sou <i>換実行</i> slator1.Tra <i>果の表示</i> Target.Text	列のセット urce := MemoS anslate; t := Translat	Source.Te	xt; ination;	
	begin //変 Trans //変 Trans //結 Memo end;	<i>換対象文字</i> slator1.Sou <i>換実行</i> slator1.Tra <i>果の表示</i> Target.Text	列のセット urce := MemoS anslate; t := Translat	Source.Te	xt; ination;	
	begin //変 Trans //変 Trans //結 Memo end;	<i>換対象文字</i> slator1.Sou <i>換実行</i> slator1.Tra <i>果の表示</i> Target.Text	列のセット urce := MemoS anslate; t := Translat	Source.Te	xt; ination;	
	begin //変 Trans //変 Trans //結 Memo end;	<i>換対象文字</i> slator1.Sou <i>換実行</i> slator1.Tra <i>果の表示</i> Target.Text	列のセット urce := MemoS anslate; t := Translat	Source.Te	xt; ination;	
	egin //変 Trans //変 Trans //結 Memo nd;	<i>換対象文字</i> slator1.Sou <i>換実行</i> slator1.Tra <i>果の表示</i> Target.Text	列のセット urce := MemoS anslate; t := Translat	Source.Te	xt; ination;	