

Windowsからスマホまでマルチデバイスでサクサク外部API呼び出し活用法

第33回 エンバカデロ・デベロッパークャンプ

エンバカデロ・テクノロジーズ
毛利 春幸



embarcadero®
DEVELOPER CAMP

■はじめに

- 現在、多くの企業がより新しい価値のあるサービス構築を目指して、多数の Web API を公開し、第 3 者企業とサービス連携を行っています。
- API 公開によってサービスの応用範囲や収益性が向上する点は、Google、Twitter、Facebookといった企業によって証明されており、すでに多くの企業にとっては API を公開することは当たり前になっています。
- Delphi / C++Builderは RESTを得意とするコンポーネントが揃っています APIをご紹介しながら 各コンポーネントの使い方を、ご説明してまいります。



アジェンダ

- 各社REST サービスAPIについて
 - Amazon API Gateway
- OAuth 2.0 の仕組みと認証方法
 - OAuth2Authenticatorコンポーネントを使って簡単OAuth2.0
 - Line, Yahoo, docomo IDなどの認証
- AWS
 - S3
- Kinveyを使ったiOSプッシュ通知
- 駅すばあと API 接続
 - 位置情報から最寄駅検索



■ 各社REST サービスAPIについて

- REST Service API



REST

- Representational State Transfer (REST) は、ウェブのような分散ハイパーメディアシステムのためのソフトウェアアーキテクチャのスタイルのひとつである。
(インターネットのプロトコルの中心となったHTTP/Web技術がベースです)
- JSON/ XMLなどのフォーマットを用いてデータのやりとりを行います
これからご説明する「Amazon Web Service」や「LINE」「Kinvey」「駅すぱあと」はAPIサービスをXML / JSONで提供しています
- その他、今回説明しないRESTサービスとしてはdocomo ID, Yahoo ID, Google, Facebook, spotifyなどが有名です



各社REST サービスAPIについて

- OAuth 2.0
 - ID+パスワードで外部サービスを利用する
Google+, Line, YahooID, FacebookなどのID認証されたユーザーは
自社サービスを許可とする、認証代行のようなイメージ
- ストレージサービス
 - AWS S3, Azure ストレージアカウント(BLOBストレージ)などのクラウドストレージ提供
- 独自データ提供
 - 天気予報, 路線情報(駅すぱあと), 番組表, 音楽情報(spotify)



■ Amazon API Gateway



Amazon API Gateway

- RESTful なアプリケーションプログラミングインターフェイス (API) の作成、デプロイ、および管理によるバックエンド HTTP エンドポイント、AWS Lambda 関数、またはその他の AWS サービスの公開をサポートし、フロントエンド HTTP エンドポイントを介して、公開された API メソッドを呼び出す AWS サービス。



Amazon API Gateway

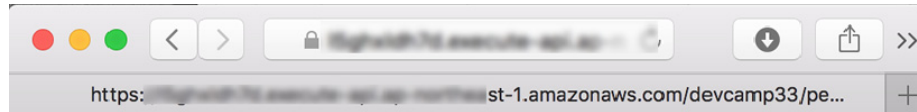
- API Gateway のサンプルAPIを設定
 - サンプル「PetStore」 API作成

The screenshot shows the Amazon API Gateway console interface. On the left, the 'API' section is expanded, showing the 'PetStore' API. A red arrow points from the 'PetStore' resource in the left sidebar to the 'GET' method in the main panel. The main panel shows the 'リソース' (Resources) section with a tree view: '/' (GET, OPTIONS, POST), '/pets' (GET, OPTIONS, POST), and '/{petId}' (GET, OPTIONS). The 'GET' method under '/pets' is highlighted with a red box. Below this, the 'メソッド' (Methods) section shows the configuration for the selected 'GET' method, including 'Mock エンドポイント' (Mock Endpoint) and 'API キー' (API Key) settings. The text 'ここにアクセス' (Access here) is written in red below the resource list.



Amazon API Gateway

- API Gateway 「PetStore (サンプル)」 ブラウザで参照
 - https://*****execute-api.ap-northeast-1.amazonaws.com/devcamp33/pets

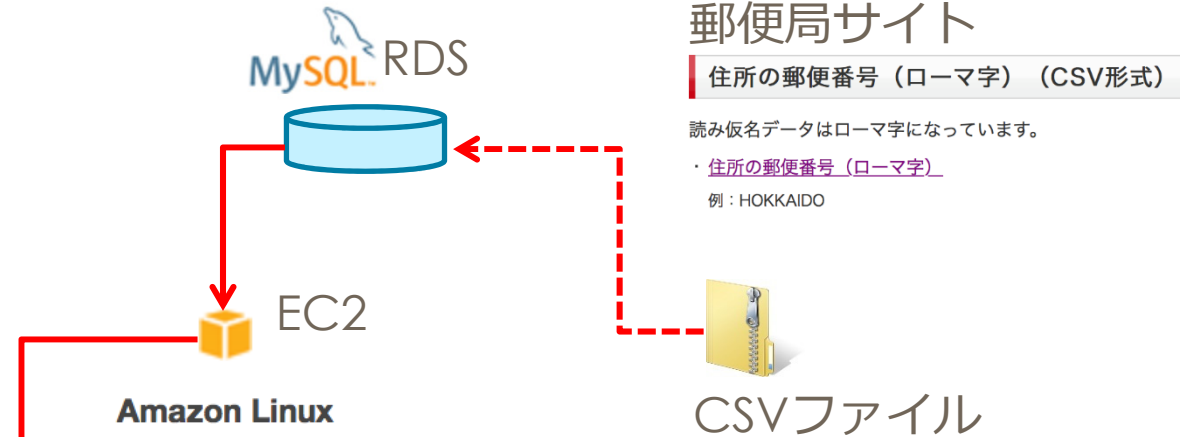


```
[
  {
    "id": 1,
    "type": "dog",
    "price": 249.99
  },
  {
    "id": 2,
    "type": "cat",
    "price": 124.99
  },
  {
    "id": 3,
    "type": "fish",
    "price": 0.99
  }
]
```



Amazon API Gateway

- API Gatewayを少しカスタマイズ



<http://www.post.japanpost.jp/zipcode/download.html>



Amazon API Gateway

- API Gateway 「v1/yubin/」 Delphi/C++Builderから接続
 - `https://*.execute-api.ap-northeast-1.amazonaws.com/devcamp33/v1/yubin/`



Amazon API Gateway

- API Gateway 「v1/yubin/」 Delphiから接続
 - devcamp33/v1/yubin/ を取得する

```
procedure TForm1.Button1Click(Sender: TObject);           //検索ボタンイベント
var
  p1: TRESTRequestParameter;
begin
  RESTClient1.BaseURL := 'https://*execute-api.ap-northeast-1.amazonaws.com/devcamp33';
  RESTRequest1.Resource := 'v1/yubin/';                 //リソース
  RESTRequest1.Params.Clear;
  p1 := RESTRequest1.Params.AddItem;
  p1.name := 'zip_num';                                  //郵便番号パラメータ
  p1.Value := Edit1.Text;
  RESTRequest1.Execute;                                  //REST実行

  MemTableSet(RESTResponse1.JSONValue as TJSONArray);   //TJSONArrayが取得できる
end;
```



Amazon API Gateway

- API Gateway 「v1/yubin/」 C++Builderから接続
 - devcamp33/v1/yubin/ を取得する

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    RESTClient1->BaseURL      = "https://*.execute-api.ap-northeast-1.amazonaws.com/devcamp33";
    RESTRequest1->Resource    = "v1/yubin/";
    RESTRequest1->Params->Clear();
    auto p1                   = RESTRequest1->Params->AddItem();
    p1->name                   = "zip_num";
    p1->Value                   = Edit1->Text;
    RESTRequest1->Execute();

    MemTableSet(static_cast<TJSONArray*>( RESTResponse1->JSONValue ) );
}
```



Amazon API Gateway

- API Gateway 「v1/yubin/」 Delphiから接続
 - devcamp33/v1/yubin/ でJSON取得後 TFDMemTableに登録

```
const field_names: array[0..3] of string = ('zip_num', 'pref', 'city', 'address1');

procedure TForm1.MemTableSet(const ajson: TJSONArray);
var
  i: Integer; field_name: String; a_item: TListViewItem; a_line: String;
begin
  FDMemTable1.Open;
  for i := 0 to Pred(ajson.Count) do
    begin
      FDMemTable1.Append;
      FDMemTable1.FieldName('zip_uniq').AsInteger :=
        StrToInt(ajson.Items[i].GetValue<TJSONString>('zip_uniq').value);
      for field_name in field_names do
        begin
          FDMemTable1.FieldName(field_name).AsString :=
            ajson.Items[i].GetValue<TJSONString>(field_name).value;
        end;
      end;
    FDMemTable1.Post;
  end;
```

//FDMemTableの初期設定は済前提
//TJSONArrayデータをコピー

//フィールド配列順でコピー



Amazon API Gateway

- API Gateway 「v1/yubin/」 C++Builderから接続
 - devcamp33/v1/yubin/ でJSON取得後 TFDMemTableに登録

```
std::vector<UnicodeString> field_names{"zip_num","pref","city","address1"};
void __fastcall TForm1::MemTableSet(TJSONArray* ajson)
{
    TJSONObject* a_buf{nullptr};
    FDMemTable1->Open(); //FDMemTableの初期設定は済前提
    for (int i = 0; i < ajson->Count; i++) //TJSONArrayデータをコピー
    {
        a_buf = static_cast<TJSONObject*>(ajson->Get(i));

        FDMemTable1->Append();
        for (auto field_name : field_names) //フィールド配列順でコピー
        {
            FDMemTable1->FieldByName(field_name)->AsString =
                static_cast<TJSONString*>(a_buf->GetValue(field_name) )->Value();
        }
    }
    FDMemTable1->Post();
}
```



■ OAuth 2.0 の仕組みと認証方法

認可フレームワーク



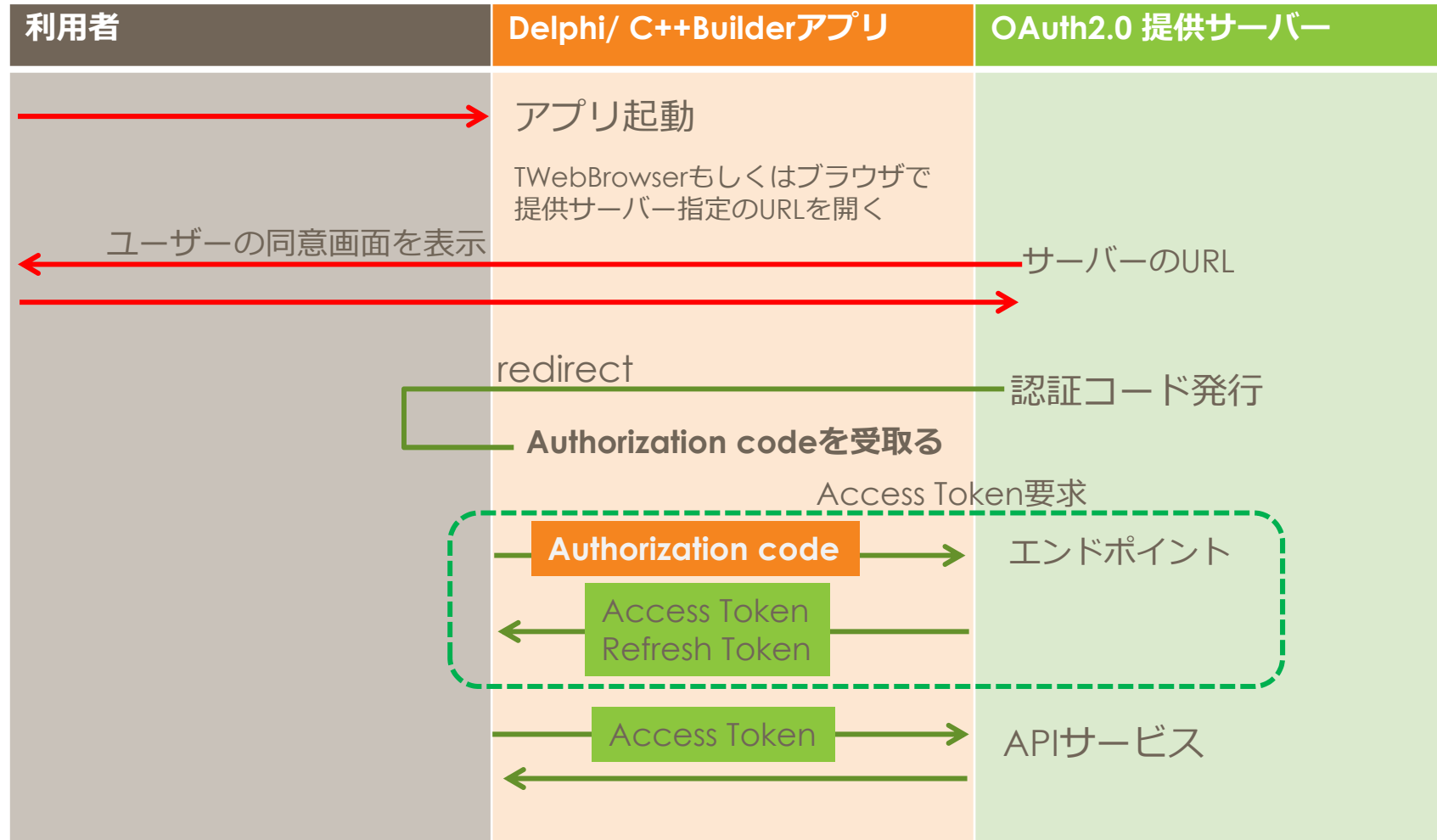
The OAuth 2.0 Authorization Framework

- OAuth 2.0 は, サードパーティーアプリケーションによるHTTPサービスへの限定的なアクセスを可能にする認可フレームワークである。
サードパーティーアプリケーションによるアクセス権の取得には, リソースオーナーとHTTPサービスの間で同意のためのインタラクションを伴う場合もあるが, サードパーティーアプリケーション自身が自らの権限においてアクセスを許可する場合もある。
- OAuth 2.0はRFC 5849に記載されているOAuth 1.0 プロトコルを廃止し, その代替となるものである。

(<https://openid-foundation-japan.github.io/rfc6749.ja.html>)



OAuth 2.0 の仕組み



OAuth 2.0 「LINE Login」

- LINE Login機能をサービスに組み込むことにより、ユーザーがLINEで簡単にサービスにログインすることが可能となります。
- ログインしたユーザーから「表示名」、「プロフィール画像」と「ひとつこと」を取得することができるため、ユーザーに登録していただく作業が必要なくなります。

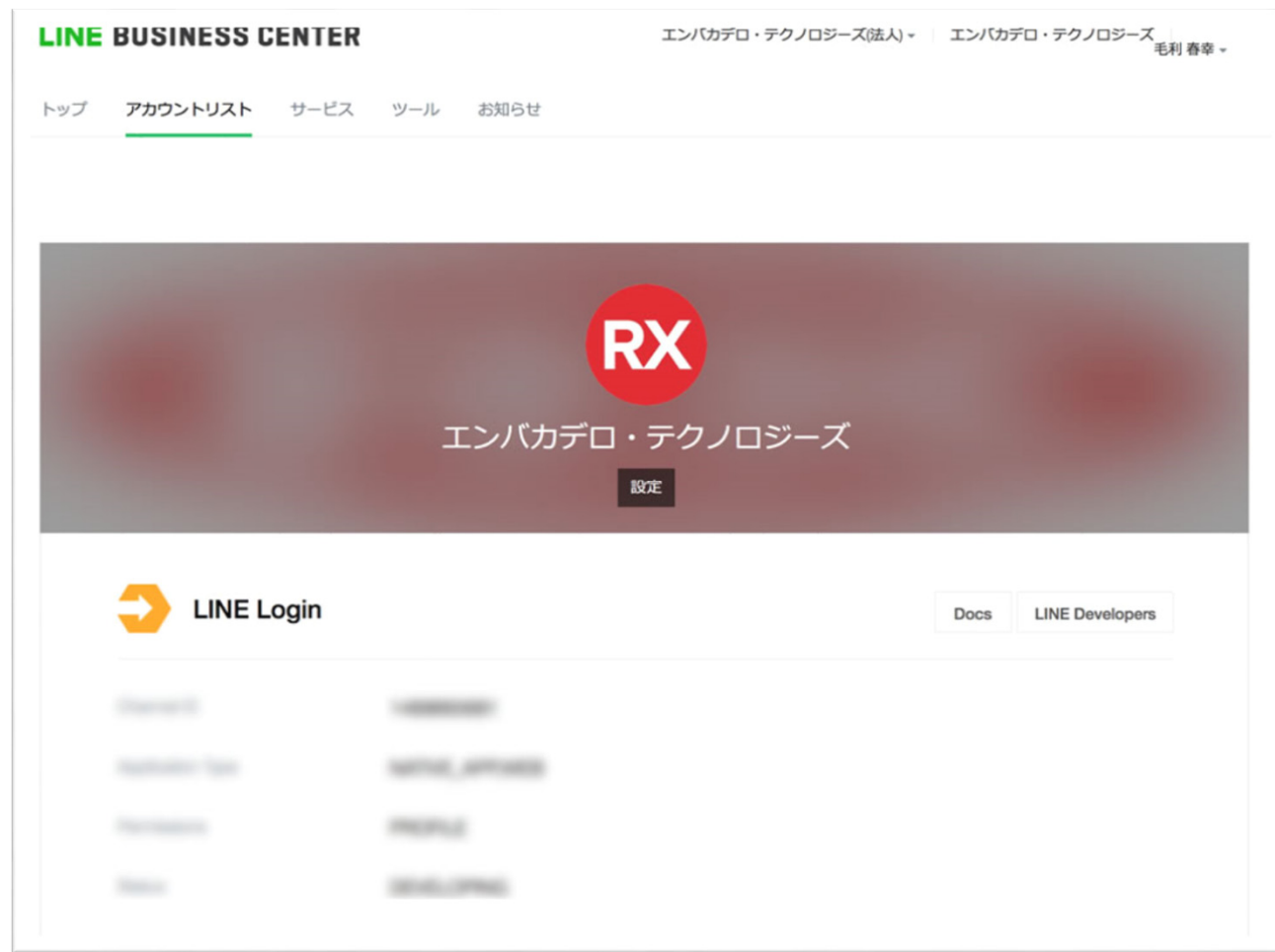


LINE Login




OAuth 2.0 「LINE Login」

- LINE Login
企業登録が必要です



OAuth 2.0 「LINE Login」

- LINE Login企業登録するとChannel IDとChannel Secretが取得できます



The screenshot shows the LINE developers console interface. At the top left, there is a green header with 'LINE developers' and a dark grey dropdown menu with the 'RX' logo and the text 'エンバカデロ・テクノロジーズ'. The main content area displays the channel name 'エンバカデロ・テクノロジーズ' in large characters. To the right of the name are links for 'History' and a green 'EDIT' button. Below the name, there are three rows of information:

Channel ID	1-4299050001
Channel Secret	●●●●●●
Name (English)	エンバカデロ・テクノロジーズ

Below the table, there is a 'SHOW' button next to the Channel Secret field. At the bottom, the 'Description (English)' is shown as 'エンバカデロ・テクノロジーズ デベロッパーキャンプ大阪イベントデモ用 RAD Studio 10.1BerlinとLINE OAuth2.0連動'.



OAuth 2.0 「LINE Login」

- 認証画面からのコールバックURLの指定が必須

NATIVE_APP

iOS Bundle ID

iOS Scheme

Android Package Name

Android Package Signature

Android Scheme

WEB

Callback URL

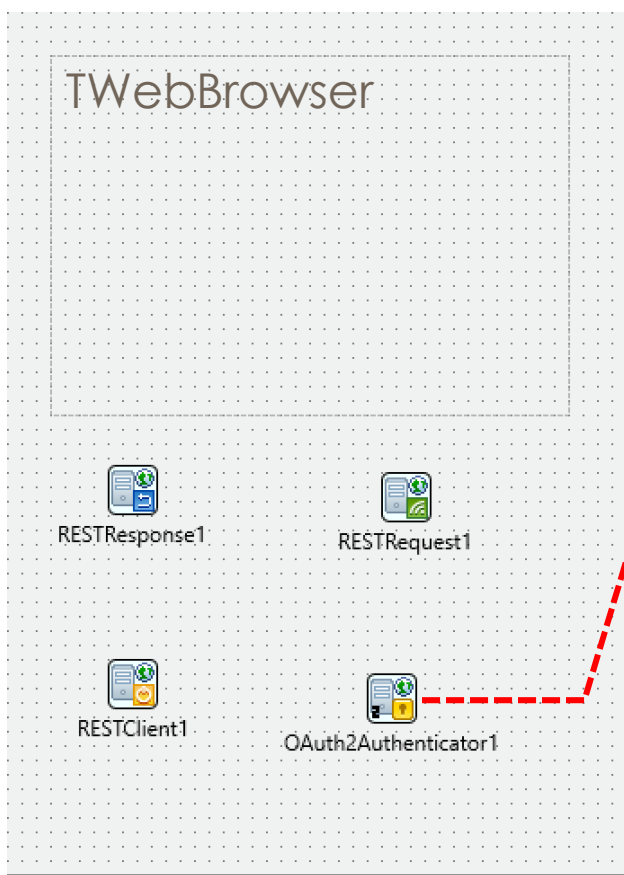
アプリ側で Schemeが設定されている場合は
iOS/Android Schemeなどの設定で良いかと思えます

今回は普通のURLコールバックにしました



OAuth 2.0 「LINE Login」

- Delphi / C++Builder から接続



OAuth2Authenticator1 TOAuth2Authenticator

検索

プロパティ イベント

AccessToken	
AccessTokenEnd	https://channel-apis.line.naver
AccessTokenExp	
AccessTokenPar	access_token
AuthCode	
AuthorizationEnd	
BindSource	OAuth2Authenticator1.BindSou
ClientID	14899623311
ClientSecret	0007c4d8b4a4d8962f18f739783974
LiveBindingデザイ	LiveBindingデザイア
LocalState	
Name	OAuth2Authenticator1
RedirectionEndp	http://localhost.com/your.php
RefreshToken	
ResponseType	rtCODE
Scope	9988
Tag	0
TokenType	ttNONE

Configure デジタルにバインド

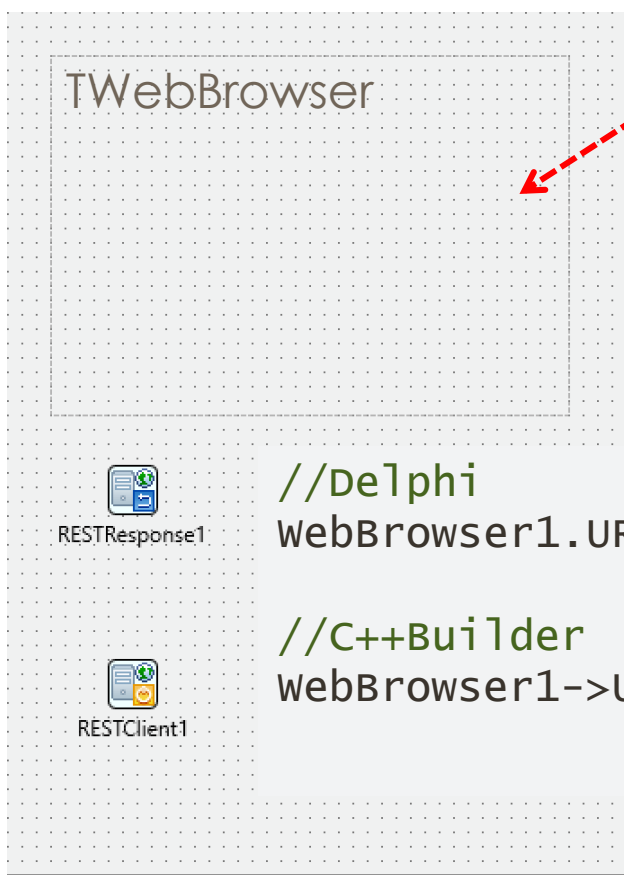
Channel ID
Channel Secret

Redirect URL



OAuth 2.0 「LINE Login」

- Delphi / C++Builder から接続



LINE developer'sの仕様通りログイン画面を起動する

Login Screen

When a user presses the LINE Login button on your website, the user is redirected to the login screen of the LINE Platform for authentication and authorization. The URL is as follows.

```
https://access.line.me/dialog/oauth/weblogin?response_type=code&client_id={Channel ID}&redirect_uri={Callback URL}&state={State}
```

For query parameters, the following must be specified.

//Delphi

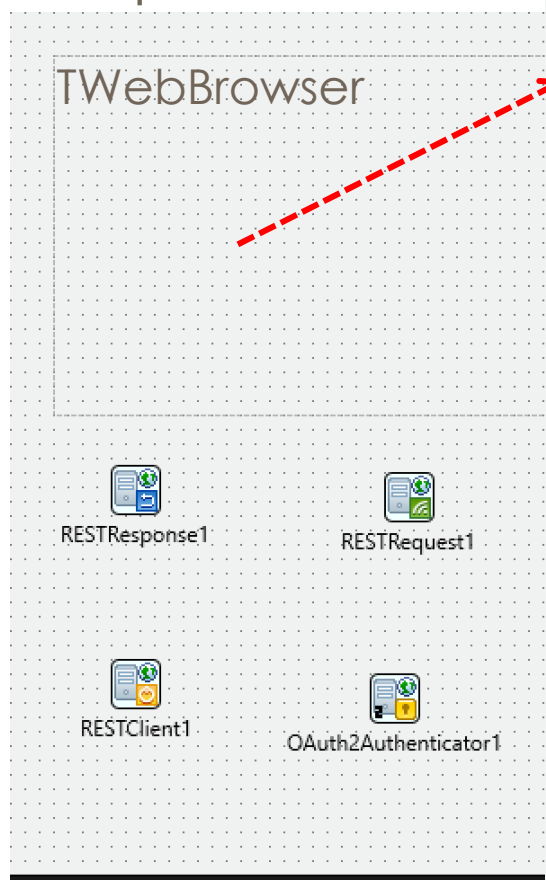
```
WebBrowser1.URL := 'https://access.line.me/dialog/oauth/weblogin....'
```

//C++Builder

```
WebBrowser1->URL = "https://access.line.me/dialog/oauth/weblogin...."
```

OAuth 2.0 「LINE Login」

- Delphi / C++Builder から接続



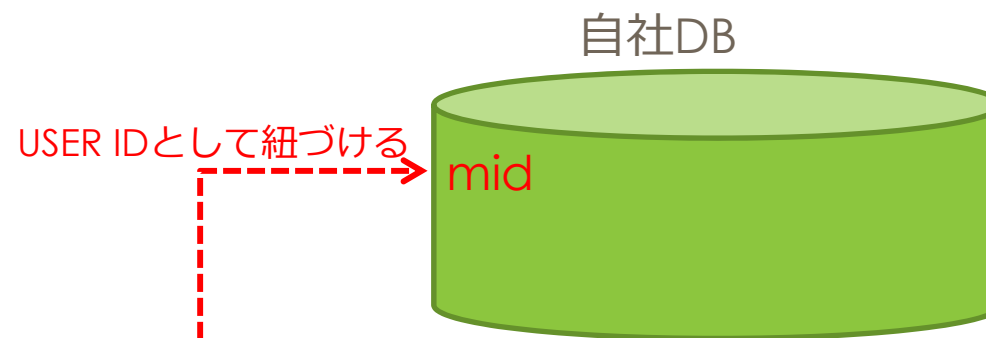
```
//Delphi
procedure TForm1.WebBrowser1DidFinishLoad(ASender: TObject);
var
  stTemp: String;
begin
  stTemp := WebBrowser1.URL;
  if Pos('code=', stTemp) > 0 then
  begin
    stTemp := Copy(stTemp, Pos('code=', stTemp)+5, Length(stTemp));
    stTemp := Copy(stTemp, 1, Pos('state=', stTemp)-2);
    OAuth2Authenticator1.AuthCode := stTemp;
    OAuth2Authenticator1.ChangeAuthCodeToAccessToken;
    if Length(OAuth2Authenticator1.AccessToken) > 0 then
    begin
      RESTRequest1.Method := TRESTRequestMethod.rmGET;
      RESTClient1.BaseURL := 'https://api.line.me/v1/profile';
      RESTRequest1.Execute; //AccessTokenが取れたのでprofile参照可能
    end;end;end;
```

OAuth 2.0 「LINE Login」

■ LINE Loginから取得したJSON

```
{
  "displayName": "Haruyuki",
  "mid": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
  "pictureUrl": "http://dl.profile.line-cdn.net/xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx75c9",
  "statusMessage": "Nexus6に変更"
}
```

```
//JSON
{
  "displayName": "Haruyuki",
  "mid": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //←ここがユニーク
  "pictureUrl": "http://dl.profile.line-cdn.net/xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
  "statusMessage": "Nexus6に変更"
}
```



ID パスワード管理などはLINEに任せてしまう事が可能になります



■ Kinveyを使ったiOSプッシュ通知

- Kinveyとはアメリカの企業であるKinvey社が提供しているBaaS(Backend as a Service)です。



Kinvey

- Kinveyとはアメリカの企業であるKinvey社が提供しているBaaS(Backend as a Service)です。
 - Users ユーザー管理
 - ストレージ
 - Engagement プッシュ通知
- メリット
 - サーバーを社内で構築しメンテナンスが必要ない

プッシュ通知デモの前に少しだけKinvey機能紹介致します



Kinvey : Users ユーザー管理

- Users ユーザー管理
 - ユーザー追加、削除などブラウザから行う事ができます

MOHRI Development

Users

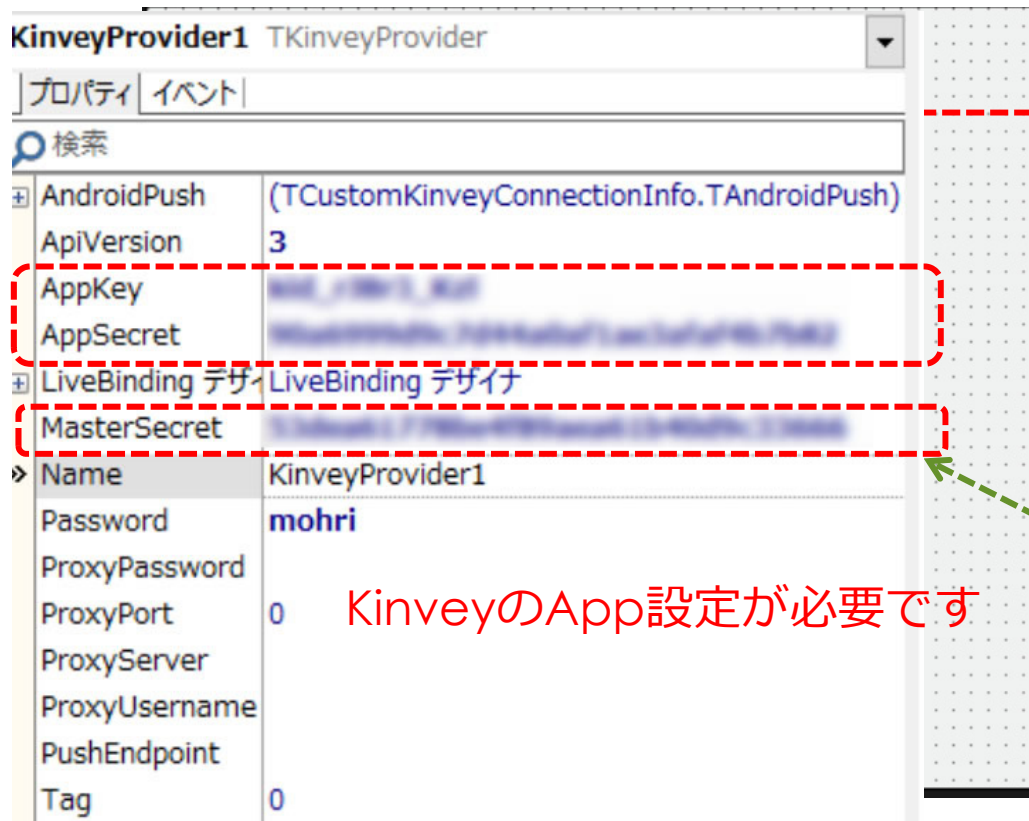
+ Column 👁 ⚙ 👤 Add User 👤 ↻ 1 user

_id	_acl	_kmd	username
583fa2d03176...	{\"creator\": \"583fa2...	{\"lmt\": \"2016-12...	\"mohri\"

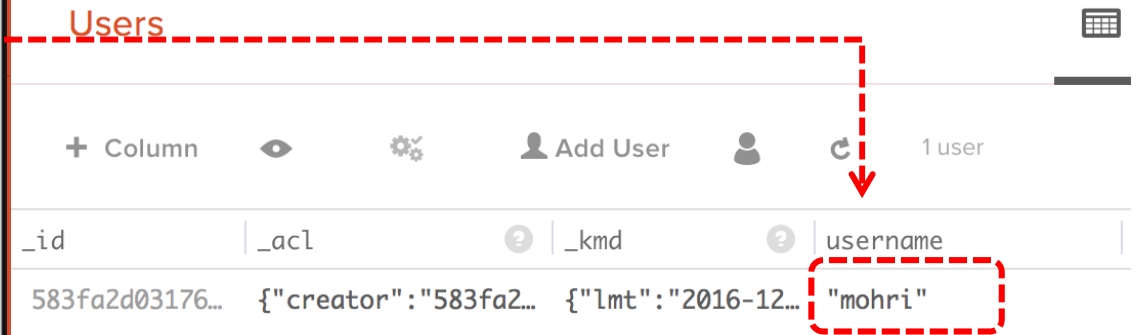


Kinvey : Delphi/C++Builderからのユーザーログイン

- TKinveyProvider + TBackendAuth を使ってアプリのユーザー認証を行います



KinveyのApp設定が必要です



ユーザーを確認

ユーザーが無い場合は EHTTPProtocolException



Kinvey : Delphiからのユーザーログアウト

- TKinveyProvider + TBackendAuth を使ってアプリのユーザー認証を行います

```
//Delphi
procedure TForm1.Button2Click(Sender: TObject);
begin
  try
    BackendAuth1.Logout;          //ログアウト
    Memo1.Lines.Append('BackendAuth1.Logout; OK' );
  except
    on e: exception do
      Memo1.Lines.Append('BackendAuth1.Logout; '+ e.Message );
  end;
end;
```



Kinvey : C++Builderからのユーザーログアウト

- TKinveyProvider + TBackendAuth を使ってアプリのユーザー認証を行います

```
void __fastcall TForm1::Button2Click(TObject *Sender){
    UnicodeString s;
    try
    {
        BackendAuth1->Logout();           //ログアウト
        s = "BackendAuth1->Logout(); OK";
    }
    catch(EHTTPProtocolException* e1)
    {
        s = "BackendAuth1->Logout(); " + e1->Message;
    }
    Memo1->Lines->Append(s);
}
```



Kinvey : ストレージ

- データ保管場所を提供しています

The screenshot displays the Kinvey Development console. The left sidebar contains navigation items: Dashboard, Users, Data (highlighted with a dashed yellow box), Data Links, and Business Logic. The top right shows the environment 'MOHRI Development' and 'Devcamp33'. Below the navigation, there are controls for '+ Row', '+ Column', an eye icon, and a gear icon. A table header is visible with columns labeled '_id' and '_acl'.



Kinvey : ストレージ

- TBackendStorageを使って 保存や取り出しが可能です

The image shows a Kinvey console interface. On the left, a configuration panel for 'BackendStorage1' and 'KinveyProvider1' is visible, with buttons for '新規作成' (New) and '更新' (Update). Below these are input fields for '文字列テスト' (String Test) and a text area containing '안녕하세요 Сайн байна уу'.

In the center, a red sidebar menu lists navigation options: Dashboard, Users, Data, Data Links, and Business Logic.

On the right, a data table is displayed with the following structure:

_id	_acl	_kmd	title_name	memo_content
5840c6...	{"creator":"kid_rJ..."}	{"lmt":"2016-12-02..."}	"文字列テスト"	"안녕하세요 Сайн байна уу"

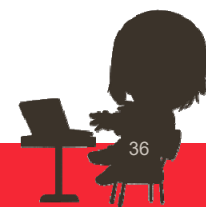
Below the table, there are controls for 'ROW' and 'COLUMN' views, and a search icon.



Kinvey : ストレージ

- Delphi + TBackendStorageを使ったクラス保存コード

```
//Delphi
FBackendList: TBackendObjectList<TKinvey_memo>;
var
  obj_: TKinvey_memo;
  entry_: TBackendEntityValue;
begin
  obj_ := TKinvey_memo.Create;
  obj_.title_name := Edit1.Text;
  obj_.memo_content := Memo1.Lines.Text;
  BackendStorage1.Storage.CreateObject<TKinvey_memo>('Devcamp33', obj_, entry_);
  FBackendList.Add(obj_, entry_);
end;
```



Kinvey : プッシュ通知

■ プッシュ通知送信方法

MOHRI Development

Push Branding Analytics

Send Push Configuration

Send a push notification to all users:

[+ add condition ...](#) [advanced ...](#)

メッセージを入力し「SEND PUSH」

Message

Don't forget about the sale today! 20% off purchases over \$20!

SEND PUSH

Recent Notifications

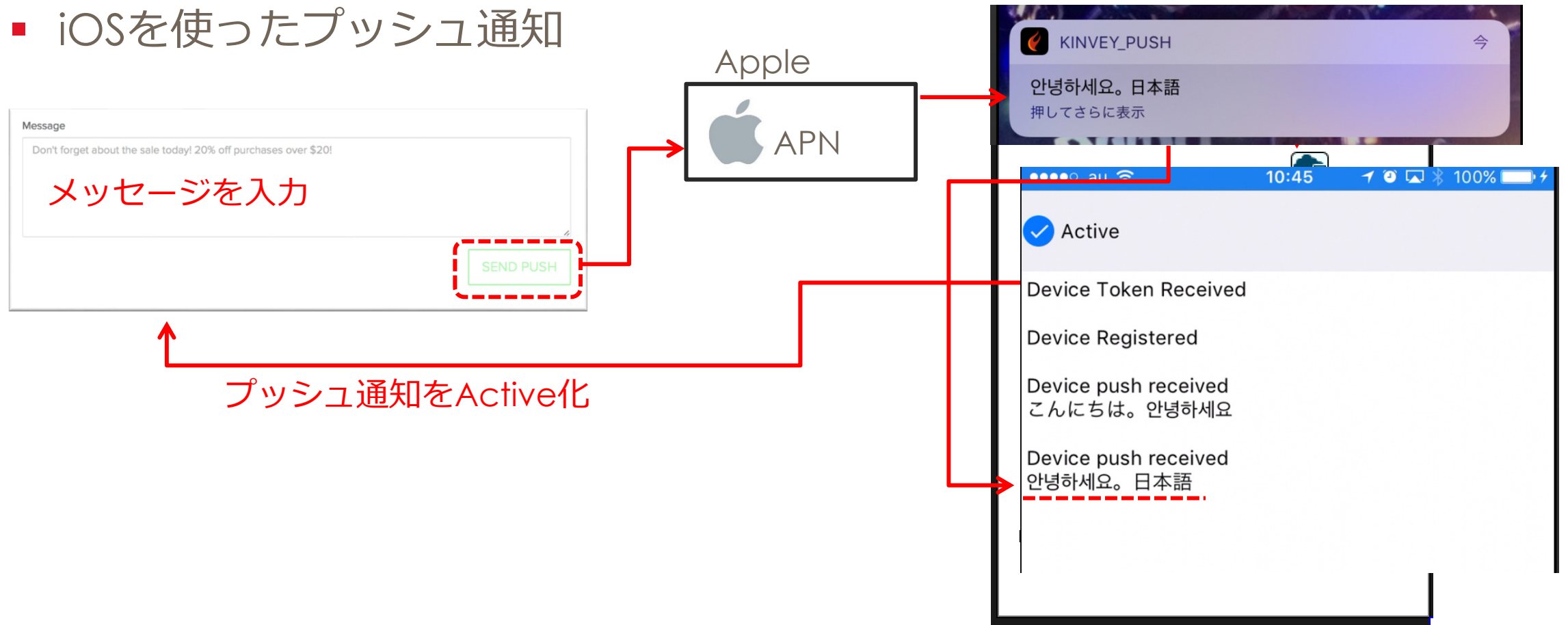
Time	Description	Sent To	Status
------	-------------	---------	--------

10:10
mohri



Kinvey : プッシュ通知

■ iOSを使ったプッシュ通知



■ AWS S3接続

- Amazon Simple Storage Service はインターネット用のストレージサービスです。
- ウェブスケールのコンピューティングを開発者が簡単に利用できるよう設計されています。



AWS S3接続

- Delphi/C++ Builderは AWS接続のための TAmazonConnectionInfoコンポーネントが用意されています



AmazonConnectionInfo1 TAmazonConnectionInfo

検索

プロパティ イベント

AccountKey	AKIAJN6H9W9W4TQBP0MA
AccountName	AKIAJN6H9W9W4TQBP0MA
ConsistentRead	<input checked="" type="checkbox"/> True
LiveBinding デザイン	LiveBinding デザイン
MFAAuthCode	
MFAserialNumber	
Name	AmazonConnectionInfo1
Protocol	https
QueueEndpoint	queue.amazonaws.com
RequestProxyHost	
RequestProxyPort	0
StorageEndpoint	s3-ap-northeast-1.amazonaws.com
TableEndpoint	sdb.amazonaws.com
Tag	0
UseDefaultEndpoints	<input type="checkbox"/> False

IAM (Identity and Access Management)

ユーザー: embarcadero

ユーザーの ARN `arn:aws:iam::111111111111:user/embarcadero`

パス /

作成時刻 2016-11-11 08:02:17CST-0900

アクセス権限 グループ (1) 認証情報 アクセスアドバイザー

アクセス権限の追加

s3_embarcadero - グループ embarcadero の 管理ポリシー



AWS S3接続

■ Delphi/C++Builderを使ってS3接続

//少し省略しています

```
std::unique_ptr<TStringList> list_{std::make_unique<TStringList>(new TStringList()) };
std::unique_ptr<TAmazonStorageService> s3{new TAmazonStorageService(AmazonConnectionInfo1)};
TCloudResponseInfo* res_{new TCloudResponseInfo()};
try{
    TAmazonBucketResult* bs = s3->GetBucket("embarcadero-jp", list_.get(),
                                             res_,TAmazonRegion::amzrAPNortheast1);

    std::unique_ptr<TMemoryStream> mm{new TMemoryStream()};
    mm->LoadFromFile("01.jpg");
    TArray<Byte> arr1;
    arr1.Length = mm->Size;
    mm->ReadBuffer(arr1, mm->Size);
    mm->Position = 0;
    //ファイルをアップロードする
    s3->UploadObject("embarcadero-jp", "01.jpg", arr1, true, nullptr, nullptr,
                    TAmazonACLType::amzbaPublicRead, res_);

    mm->Clear();
    //ファイルを取得する
    s3->GetObject( "embarcadero-jp", "01.jpg", mm.get(), res_);
```



■ 駅すぱあとWebサービス API 接続

- 鉄道情報を商用利用可能なAPI



駅すばあとWebサービス フリープラン

- 信頼の鉄道情報を商用利用可能なAPIを0円で提供
 - <https://ekiworld.net/>



The screenshot shows the EkiWorld website interface. At the top, there is a navigation bar with the EkiWorld logo, a search bar, and links for FAQ, contact, and online shop. Below the navigation bar, there are tabs for 'About EkiWorld', 'Services & Product Introduction', 'Terms of Use', and 'Support'. The main content area features a large banner with the text '0円から利用できるクラウド型API 駅すばあとWebサービス フリープラン' and a sub-headline '「駅すばあと」が集めた「駅情報」や「路線情報」をオープンデータ化！'. Below the banner, there is a download icon and a paragraph of text. At the bottom of the banner, there are two buttons: '「フリープラン」のお申し込み' (Apply for Free Plan) and '「スタンダードプラン」評価版' (Standard Plan Evaluation Version). A green dashed arrow points from the 'Apply for Free Plan' button to the text below.



アクセスキー、ドキュメント、サンプルコードなどが送られてきます



駅すぱあとWebサービス フリープラン + C++Builder

- 駅すぱあと Webサービス フリープランAPI

項目	エンドポイント
駅付加情報	/v1/*/station/info
会社情報	/v1/*/corporation
経路探索	/v1/*/search/course/light
路線情報	/v1/*/rail
線区情報	/v1/*/line
駅情報	/v1/*/station
駅簡易情報	/v1/*/station/light



駅すぱあとWebサービス フリープラン + C++Builder

```
enum TEki_apis_ {info, corporation, search_course_light, rail, line, station, station_light};
class TEki_Api : TObject
{
    (省略)
    std::map<TEki_apis_, String> FEndPoint{
        {TEki_apis_::info,          "/station/info"},
        {TEki_apis_::corporation,   "/corporation"},
        {TEki_apis_::search_course_light, "/search/course/light"},
        {TEki_apis_::rail,          "/rail"},
        {TEki_apis_::line,          "/line"},
        {TEki_apis_::station,       "/station"},
        {TEki_apis_::station_light, "/station/light"}};


    (省略)
    __property UnicodeString EkiKey           //アクセスキー
    __property TEki_apis_ ApiChannel         //エンドポイント選択
    __property UnicodeString EndPoint        //エンドポイント名(ReadOnly)
    __property UnicodeString EkiName         //駅名
    __property TNotifyEvent OnHttpJson      //イベント
    __property UnicodeString EkiTo          //駅 TO
    __property UnicodeString EkiFrom        //駅 FROM
    __property int EkiCode                  //駅 コード
}
```



駅すぱあとWebサービス フリープラン + C++Builder

Form1

駅名入力



```
void __fastcall TForm1::OnEditChange(TObject *Sender)
{
    ListView1->Items->Clear();
    TEdit* edit_      = static_cast<TEdit*>(Sender);
    FselEdit          = edit_;
    ListView1->Tag    = edit_->Tag;
    FEki_api->ApiChannel  = TEki_apis_::station;
    FEki_api->EkiName    = edit_->Text;      //入力した駅名
    FEki_api->OnHttpJson = DoEditStation;  //イベント
    FEki_api->UpdateExcect();              //API実行
}
```



駅すぱあとWebサービス フリープラン + C++Builder

■ JSON REST APIをC++Builderで呼び出す

Form1

出発駅
東京 駅情報

目的地
江戸 駅情報

江戸橋 ≡

江戸川 ≡

江戸川橋 ≡

江戸川台 ≡

駅情報 出力

- 各ホーム⇄改札内南通路
- 京葉線ホーム⇄改札
- 総武線ホーム⇄丸の内地下改札
- 八重洲連絡通路⇄京葉地下八重洲口改札内
- 改札内中央通路⇄B1F 駅の鈴広場
- 丸の内地下北口改札外⇄丸の内北口交番付近
- 京葉地下丸の内口改札⇄4番出口
- 京葉地下丸の内口改札⇄6番出口

【東京メトロ】

- ホーム⇄改札
- 改札⇄1番出口 (オアゾ内)
- 改札⇄地上 (丸ビル方面) {"Name": "エスカレーター", "Comment": "【J R】"}

【新幹線】

- 各ホーム⇄各改札

【在来線】

- 各ホーム⇄各改札

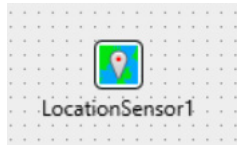
Debug Out

```
max=7  
{"apiVersion": "1.27.0.0", "max": "7", "offset": "1", "engineVersion": "2011"}  
"東京"  
"22828"  
"成田空港(東京)"  
"29110"  
"羽田空港(東京)"  
"22827"  
"とうきょうスカイツリー"  
"22859"  
"東京レポート"  
"29073"  
"東京ディズニーシー・ステーション"  
"29372"  
"東京ディズニーランド・ステーション"  
"29370"  
max=1
```



駅すぱあと路線図 フリープラン + C++Builder

- Rosen JS (JavaScript)で作られたブラウザAPI
- C++Builder接続の場合
 - ロケーションセンサー(Windows, Mac, iOS, AndroidのGPS機能をカプセル化したコンポーネント)を使い位置情報取得



- TWebBrowser(ブラウザ)からJSの関数をキックする
 - Delphi/C++Builderからブラウザ内のJavaScriptを操作可能です



まとめ

- 各社REST サービスAPIについて
 - Amazon API Gateway
- OAuth 2.0 の仕組みと認証方法
 - TOAuth2Authenticatorコンポーネントを使って簡単OAuth2.0
 - Line, Yahoo, docomo IDなどの認証
- AWS
 - S3
- Kinveyを使ったiOSプッシュ通知
- 駅すばあと API 接続
 - 位置情報から最寄駅検索



THANKS!

www.embarcadero.com/jp

第33回 エンバカデロ・デベロッパーキャンプ