日日是Oracle APEX

Oracle APEXを使った作業をしていて、気の付いたところを忘れないようにメモをとります。

2024年1月31日水曜日

Microsoft Azure AI servicesの音声サービスを呼び出してテキスト読み上げを行う

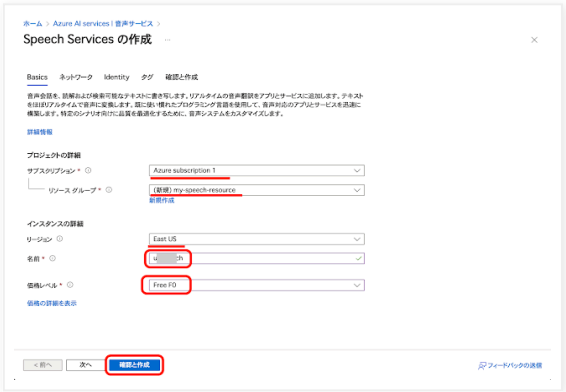
Microsoft Azure AI servicesの音声サービスのREST APIを呼び出して、テキスト読み上げを実行してみます。

こちらの記事「OpenAIのText to speech APIを呼び出して写真の説明を読み上げる」で作成したアプリケーションを、Google Gemini Pro VisionとAzure AI servicesの音声サービスの組み合わせに置き換えます。呼び出しているサービスを入れ替えるだけなので、元記事にあるアプリケーションの動作は変わりません。

Microsoft Azure側の準備を行います。

AzureのPortalより、Azure AI servicesの**音声サービス**を開き、**音声サービスの作成**を行います。

サブスクリプション、**リソース・グループ**、**リージョン**、**名前**、**価格レベル**（Free F0を選択しました）を環境に合わせて指定し、**確認と作成**をクリックします。

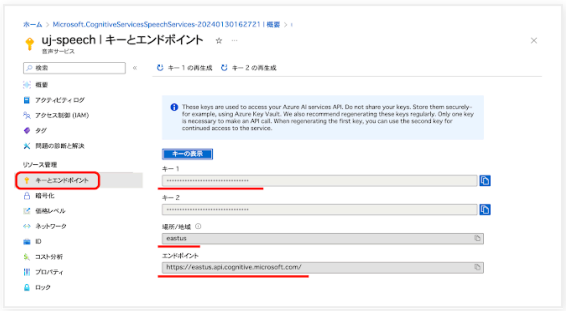


確認画面が表示されるので、**作成**をクリックします。

デプロイが完了したら、**リソースに移動**します。

サイド・メニューから**キーとエンドポイント**を開きます。

Oracle APEXからREST APIを発行する際に使用するので、**キー１**、**場所/地域**、**エンドポイント**の値をコピーして保管しておきます。

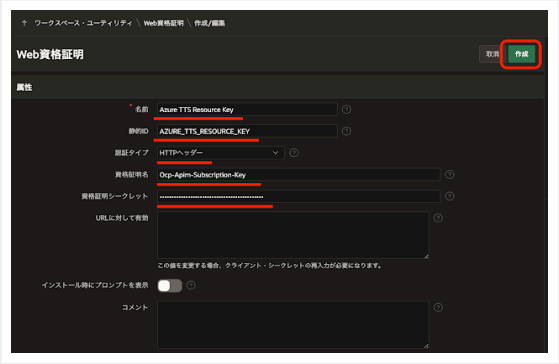


Azure側の準備は以上で完了です。

APEX側で**Web資格証明**を作成します。

最初にHTTPヘッダー**Ocp-Apim-Subscription-Key**の値を**APIキー**とした**Web資格証明**を作成します。

名前は**Azure TTS Resource Key**、**静的ID**は**AZURE_TTS_RESOURCE_KEY**とします。認証タイプはHTTPヘッダー、**資格証明名**はヘッダー名の**Ocp-Apim-Subscription-Key**、**資格証明シークレット**に**APIキー**（**キー１**の値）を設定します。



リソース・キーだけでも良いのですがトークンでも認証できるように、もうひとつトークンを保持するための**Web資格証明**を作成します。

名前は**Azure TTS Bearer Token**、**静的ID**は**AZURE_TTS_BEARER_TOKEN**とします。**認証タイプ**は**HTTPヘッダー**、**資格証明名**は**Authorization**とします。

資格証明シークレットは後でコードから更新するため、**適当な文字列**を指定します。

Microsoft Azureの音声サービスを呼び出すパッケージを**UTL_AZURE_AI_SERVICES**として作成しています。以下のコードを実行し、パッケージを作成します。音声サービスの音声出力は音声は日本語のみ、NonStreamingのみを対象にしています。

```
create or replace package utl_azure_ai_services
as
/* Text to Speech Endpoints */
C_TOKEN_URL constant varchar2(80) := 'https://<REGION>.api.cognitive.microsoft.com/sts/v1.0/iss
C_LIST_URL  constant varchar2(80) := 'https://<REGION>.tts.speech.microsoft.com/cognitiveservic
C_VOICE_URL constant varchar2(80) := 'https://<REGION>.tts.speech.microsoft.com/cognitiveservic

/* Voice Selection */
C_JA_JP_NANAMI_NEURAL constant varchar2(20) := 'ja-JP-NanamiNeural';
C_JA_JP_KEITA_NEURAL  constant varchar2(20) := 'ja-JP-KeitaNeural';
C_JA_JP_AOI_NEURAL    constant varchar2(20) := 'ja-JP-AoiNeural';
C_JA_JP_DAICHI_NEURAL constant varchar2(20) := 'ja-JP-DaichiNeural';
C_JA_JP_MAYU_NEURAL   constant varchar2(20) := 'ja-JP-MayuNeural';
C_JA_JP_NAOKI_NEURAL  constant varchar2(20) := 'ja-JP-NaokiNeural';
C_JA_JP_SHIORI_NEURAL constant varchar2(20) := 'ja-JP-ShioriNeural';

/* Audio Output */
```

```
C_RIFF_8KHZ_8BIT_MONO_ALAW    constant varchar2(30) := 'riff-8khz-8bit-mono-alaw';
C_RIFF_8KHZ_8BIT_MONO_MULAW   constant varchar2(30) := 'riff-8khz-8bit-mono-mulaw';
C_RIFF_8KHZ_16BIT_MONO_PCM    constant varchar2(30) := 'riff-8khz-16bit-mono-pcm';
C_RIFF_22050HZ_16BIT_MONO_PCM constant varchar2(30) := 'riff-22050hz-16bit-mono-pcm';
C_RIFF_24KHZ_16BIT_MONO_PCM   constant varchar2(30) := 'riff-24khz-16bit-mono-pcm';
C_RIFF_44100HZ_16BIT_MONO_PCM constant varchar2(30) := 'riff-44100hz-16bit-mono-pcm';
C_RIFF_48KHZ_16BIT_MONO_PCM   constant varchar2(30) := 'riff-48khz-16bit-mono-pcm';

/**
 * テキストを入力として、SSMLを返す。
 *
 * @param p_text      読み上げるテキスト
 * @param p_voice     音声の種類を指定
 * @param p_lang      言語の指定
 * @return SSML
 */
function text_to_ssml(
    p_text in clob
    ,p_voice in varchar2
    ,p_lang  in varchar2 default 'ja-JP'
) return clob;

/*
 * Ref: https://learn.microsoft.com/ja-jp/azure/ai-services/speech-service/rest-text-to-speech?
 */

/**
 * アクセス・トークンを取得する。
 */
function get_token(
    p_region in varchar2
    ,p_credential_static_id in varchar2
) return varchar2;

/**
 * 取得したアクセス・トークンで、Web資格証明を更新する。
 * 本当は9分間生成したアクセス・トークンをキャッシュすることが推奨されている。
 */
procedure update_token(
    p_token in varchar2
    ,p_credential_static_id in varchar2
);

/**
 * アクセス・トークンの取得と更新を一度で行う。
 */
procedure update_token(
```

```
    p_region in varchar2
    ,p_credential_header in varchar2
    ,p_credential_token  in varchar2
);

/**
 * 音声の一覧を取得する。
 */
procedure list(
    p_region in varchar2
    ,p_credential_static_id in varchar2
    ,p_response out clob
);

/**
 * テキストを音声に変換する。あらかじめ構築されたニューラル音声に変換する。
 */
procedure speech(
    p_region in varchar2
    ,p_input in clob
    ,p_voice in varchar2        default C_JA_JP_SHIORI_NEURAL
    ,p_audio_format in varchar2 default C_RIFF_24KHZ_16BIT_MONO_PCM
    ,p_user_agent   in varchar2 default 'Oracle-APEX'
    ,p_audio out blob
    ,p_credential_static_id in varchar2
);
end utl_azure_ai_services;
/

create or replace package body utl_azure_ai_services
as

/**
 * テキストをSSMLに変換する。
 *
 * Ref: Generate XML with dbms_xmldom
 * https://livesql.oracle.com/apex/livesql/file/content_DSNFKBJAVPJB7IBFKJ10UXVY2.html
 */
function text_to_ssml(
    p_text   in clob
    ,p_voice in varchar2
    ,p_lang  in varchar2 default 'ja-JP'
) return clob
as
    l_docClob CLOB;
    -- whole doc
    l_xmlDoc     dbms_xmldom.domdocument;
```

```
    l_xmlDocNode dbms_xmldom.domnode;
    l_wholeDoc   dbms_xmldom.domnode;
    -- speak element (root)
    l_speakElement     dbms_xmldom.domelement;
    l_speakElementNode dbms_xmldom.domnode;
    -- voice element
    l_voiceElement     dbms_xmldom.domelement;
    l_voiceElementNode dbms_xmldom.domnode;
    -- inner text
    l_voiceText     dbms_xmldom.domtext;
    l_voiceTextNode dbms_xmldom.domnode;
    --reusable node when appending child(s)
    l_childNode dbms_xmldom.domnode;
begin
    -- initialise the document
    l_xmlDoc := dbms_xmldom.newDOMDocument();
    dbms_xmldom.setVersion(l_xmlDoc, '1.0');
    dbms_xmldom.setCharset(l_xmlDoc, 'UTF-8');
    --convert it to a node. everything needs to be a node eventually
    l_xmlDocNode := dbms_xmldom.makeNode(
        doc => l_xmlDoc
    );
    --make a new speak element containing voice information
    l_speakElement := dbms_xmldom.createElement(
        doc => l_xmlDoc
        ,tagName => 'speak'
        ,ns => 'http://www.w3.org/2001/10/synthesis'
    );
    -- convert it to a node
    l_speakElementNode := dbms_xmldom.makeNode(
        elem => l_speakElement
    );
    /*
     * Ref: https://learn.microsoft.com/ja-jp/azure/ai-services/speech-service/speech-synthesis
     */
    dbms_xmldom.setAttribute(
        elem      => l_speakElement
        ,name     => 'version'
        ,newvalue => '1.0'
    );
    dbms_xmldom.setAttribute(
        elem      => l_speakElement
        ,name     => 'xmlns'
        ,newvalue => 'http://www.w3.org/2001/10/synthesis'
    );
    dbms_xmldom.setAttribute(
        elem      => l_speakElement
```

```
    ,name     => 'xml:lang'
    ,newvalue => p_lang
);
--make a voice element
l_voiceElement := dbms_xmldom.createElement(
    doc => l_xmlDoc,
    tagName => 'voice'
);
-- convert it to a node
l_voiceElementNode := dbms_xmldom.makeNode(
    elem => l_voiceElement
);
-- set voice
dbms_xmldom.setAttribute(
    elem      => l_voiceElement
    ,name     => 'name'
    ,newvalue => p_voice
);
/* 属性としてeffectも定義できるが、これは省略 */
--make the text element (for name)
l_voiceText := dbms_xmldom.createTextNode(
    doc => l_xmlDoc,
    data => HTF.ESCAPE_SC ( p_text )
);
--convert it to a node
l_voiceTextNode := dbms_xmldom.makeNode(
    t => l_voiceText
);
--add the voice text to the voice element
l_childNode := dbms_xmldom.appendChild(
    n => l_voiceElementNode,
    newchild => l_voiceTextNode
);
--add the voice node to the speak node
l_childNode := dbms_xmldom.appendchild(
    n => l_speakElementNode,
    newchild => l_voiceElementNode
);
--append the speak element to the document
l_wholeDoc := dbms_xmldom.appendchild(
    n => l_xmlDocNode,
    newchild => l_speakElementNode
);
--print the xml
dbms_lob.createtemporary(l_docClob, false);
dbms_xmldom.writetoclob(
    doc => l_xmlDoc,
```

```
        cl => l_docClob
    );
    return l_docClob;
end;

/**
 * アクセス・トークンの取得
 */
function get_token(
    p_region in varchar2
    ,p_credential_static_id in varchar2
) return varchar2
as
    l_token_url varchar2(80);
    l_response  varchar2(32767);
    e_api_call_failed exception;
begin
    l_token_url := replace(C_TOKEN_URL, '<REGION>', p_region);
    apex_web_service.clear_request_headers();
    apex_web_service.set_request_headers('Content-Type', 'application/x-www-form-urlencoded', p
    apex_web_service.set_request_headers('Content-Length', '0', p_reset => false);
    l_response := apex_web_service.make_rest_request(
        p_url => l_token_url
        ,p_http_method => 'POST'
        ,p_credential_static_id => p_credential_static_id
    );
    if apex_web_service.g_status_code <> 200 then
        raise e_api_call_failed;
    end if;
    return l_response;
end get_token;

/*
 * アクセス・トークンでWeb資格証明を更新する。
 */
procedure update_token(
    p_token in varchar2
    ,p_credential_static_id in varchar2
)
as
begin
    apex_credential.set_persistent_credentials(
        p_credential_static_id => p_credential_static_id
        ,p_username => 'Authorization'
        ,p_password => 'Bearer ' || p_token
    );
end update_token;
```

```
/*
 * アクセス・トークンの取得とWeb資格証明の更新を一度で行う。
 */
procedure update_token(
    p_region in varchar2
    ,p_credential_header in varchar2
    ,p_credential_token  in varchar2
)
as
    l_token varchar2(32767);
begin
    l_token := get_token(
        p_region => p_region
        ,p_credential_static_id => p_credential_header
    );
    update_token(
        p_token => l_token
        ,p_credential_static_id => p_credential_token
    );
end update_token;

/*
 * 選択可能な音声のリスト
 */
procedure list(
    p_region in varchar2
    ,p_credential_static_id in varchar2
    ,p_response out clob
)
as
    l_list_url varchar2(80);
    l_response clob;
    e_api_call_failed exception;
begin
    l_list_url := replace(C_LIST_URL, '<REGION>', p_region);
    apex_web_service.clear_request_headers();
    apex_web_service.set_request_headers('Content-Type', 'application/json', p_reset => false);
    l_response := apex_web_service.make_rest_request(
        p_url => l_list_url
        ,p_http_method => 'GET'
        ,p_credential_static_id => p_credential_static_id
    );
    if apex_web_service.g_status_code <> 200 then
        raise e_api_call_failed;
    end if;
    p_response := l_response;
```

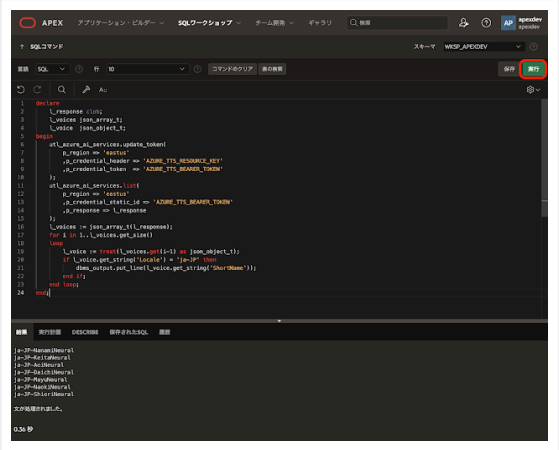
```
end list;

/*
 * Text to Speechの呼び出し。
 */
procedure speech(
    p_region in varchar2
    ,p_input in  clob
    ,p_voice in  varchar2
    ,p_audio_format in varchar2
    ,p_user_agent   in varchar2
    ,p_audio out blob
    ,p_credential_static_id in varchar2
)
as
    l_voice_url varchar2(80);
    l_request clob;
    l_request_blob blob;
    l_audio blob;
    e_api_call_failed exception;
begin
    l_voice_url := replace(C_VOICE_URL, '<REGION>', p_region);
    /* 与えられたテキストよりSSMLを生成する。 */
    l_request := text_to_ssml(
        p_text   => p_input
        ,p_voice => p_voice
    );
    apex_web_service.clear_request_headers();
    apex_web_service.set_request_headers('X-Microsoft-OutputFormat', p_audio_format, p_reset =>
    apex_web_service.set_request_headers('Content-Type', 'application/ssml+xml', p_reset => fal
    apex_web_service.set_request_headers('User-Agent', p_user_agent, p_reset => false);
    l_audio := apex_web_service.make_rest_request_b(
        p_url => l_voice_url
        ,p_http_method => 'POST'
        ,p_body => l_request
        ,p_credential_static_id => p_credential_static_id
    );
    if apex_web_service.g_status_code <> 200 then
        raise e_api_call_failed;
    end if;
    p_audio := l_audio;
end speech;

end utl_azure_ai_services;
/
```

Web資格証明が適切に作成できているか、日本語の音声一覧を取得することにより確認します。

SQLコマンドで以下のスクリプトを実行します。日本語の音声一覧が取得できれば、認証関連の設定が正しいことが確認できています。引数**p_region**に渡す値は、環境に合わせて変更します。

```
declare
    l_response clob;
    l_voices json_array_t;
    l_voice  json_object_t;
begin
    utl_azure_ai_services.update_token(
        p_region => 'eastus'
        ,p_credential_header => 'AZURE_TTS_RESOURCE_KEY'
        ,p_credential_token  => 'AZURE_TTS_BEARER_TOKEN'
    );
    utl_azure_ai_services.list(
        p_region => 'eastus'
        ,p_credential_static_id => 'AZURE_TTS_BEARER_TOKEN'
        ,p_response => l_response
    );
    l_voices := json_array_t(l_response);
    for i in 1..l_voices.get_size()
    loop
        l_voice := treat(l_voices.get(i-1) as json_object_t);
        if l_voice.get_string('Locale') = 'ja-JP' then
            dbms_output.put_line(l_voice.get_string('ShortName'));
        end if;
    end loop;
end;
```

list-supported-voices-japanese.sql hosted with ❤️ by GitHub view raw



APIの呼び出しには**Web資格証明**の**AZURE_TTS_RESOURCE_KEY**（APIキーでの認証）と**AZURE_TTS_BEARER_TOKEN**（トークンでの認証）のどちらも指定できます。トークンでの認証の場合、取得したトークンの有効期間は10分間です。Microsoftのドキュメントによると、9分間は同じトークンを使用することが推奨されています。

> **アクセス トークンを使用する方法**
>
> このサービスには、アクセス トークンを Authorization: Bearer <TOKEN> ヘッダーとして送信する必要があります。 各アクセス トークンは 10 分間有効です。 新しいトークンはいつでも取得できますが、ネットワークのトラフィックと待機時間を最小限に抑えるために、同じトークンを 9 分間使用することをお勧めします。
>
> https://learn.microsoft.com/ja-jp/azure/ai-services/speech-service/rest-text-to-speech?tabs=streaming

今回は取得したトークンの再利用を実装していないため、APIキーによる認証により音声サービスを呼び出すことにします。

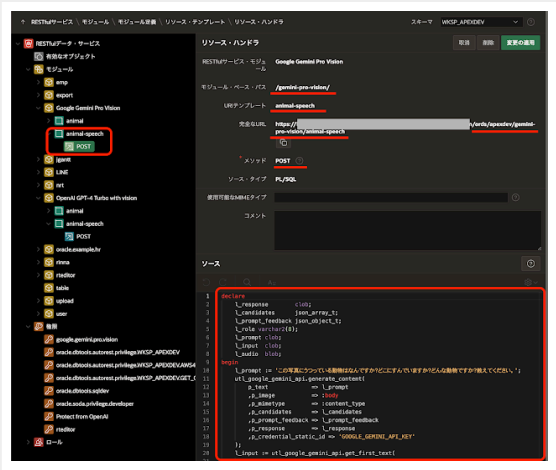
作成済みのAPEXアプリケーションに変更点はありません。

以下のコードで、写真を受け取って音声を返すRESTサービスを作成します。引数**p_region**に**eastus**を与えている部分は、環境に合わせて変更します。

```
declare
    l_response        clob;
    l_candidates      json_array_t;
    l_prompt_feedback json_object_t;
    l_role varchar2(8);
    l_prompt clob;
    l_input  clob;
    l_audio  blob;
begin
    l_prompt := 'この写真にうつっている動物はなんですか?どこにすんでいますか?どんな動物ですか?教えてください。';
    utl_google_gemini_api.generate_content(
        p_text             => l_prompt
        ,p_image           => :body
        ,p_mimetype        => :content_type
        ,p_candidates      => l_candidates
        ,p_prompt_feedback => l_prompt_feedback
        ,p_response        => l_response
        ,p_credential_static_id => 'GOOGLE_GEMINI_API_KEY'
    );
    l_input := utl_google_gemini_api.get_first_text(
        p_candidates => l_candidates
        ,p_role => l_role
    );
    utl_azure_ai_services.speech(
        p_region => 'eastus'
        ,p_input => l_input
        ,p_voice => utl_azure_ai_services.C_JA_JP_AOI_NEURAL
        ,p_audio => l_audio
        ,p_credential_static_id => 'AZURE_TTS_RESOURCE_KEY'
    );
    /* audio/pcmのファイルを返す。 */
```

```
    :status := 200;
    sys.htp.init;
    sys.htp.p('Content-Length: ' || dbms_lob.getlength(l_audio));
    sys.htp.p('Content-Type: audio/pcm');
    sys.htp.p('Content-Disposition: attachment; filename=speech.pcm');
    sys.owa_util.http_header_close;
    sys.wpg_docload.download_file(l_audio);
exception
    when others then
        htp.p(l_response);
        :status := 400;
end;
```

call-openai-gpt-then-azure-tts.sql hosted with ❤️ by GitHub view raw



アプリケーション定義の置換文字列**G_REQUEST_URL**の置換値を、新しく作成したPOSTハンドラの**完全なURL**に置き換えます。

以上で、Google Gemini ProとAzure AI servicesの音声サービスへの置き換えは完了です。

今回の記事の作業を行う本来の目的は、Azure AI servicesのREST APIをサービス・プリンシパルで認証して呼び出すことだったのですが、それはできませんでした。

Azure AI サービスに対する要求の認証
https://learn.microsoft.com/ja-jp/azure/ai-services/authentication

> **重要**
> Microsoft Entra 認証は常に、Azure リソースのカスタム サブドメイン名と共に使用される必要があります。 リージョン エンドポイントでは、Microsoft Entra 認証がサポートされていません。

また、カスタム・サブドメインはプライベートエンドポイントのみでサポートされています。

Azure AI サービスのカスタム サブドメイン名
https://learn.microsoft.com/ja-jp/azure/ai-services/cognitive-services-custom-subdomains

> **警告**
> Speech Service では、プライベート エンドポイントのみでカスタム サブドメインが使用されます。 それ以外の場合は、Speech Service および関連付けられている SDK でリージョン エンドポイントを使用してください。

Microsoft Entra IDを使用して認証する方法については、前出の「Azure AI サービスに対する要求の認証」の「Microsoft Entra IDを使用して認証する」のセクションで手順が紹介されています。トークンURLに呼び出しなどは標準的なOAuth2のクライアント資格情報フローに従っているように見えるので、Oracle APEXからもAPIアクセスに必要なトークンは取得できるでしょう。

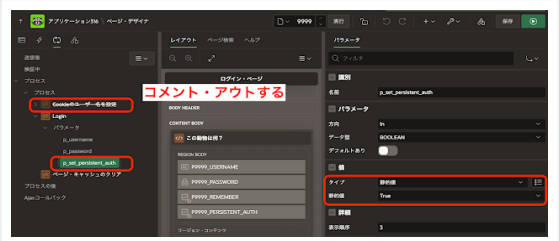
おまけ

今回作成しているアプリケーションは、PWAとしてインストールできるように設定しています。スマホなどにアプリケーションをインストールすると、アプリケーションを起動するたびにユーザー認証が要求されるため（そしてスマホ上でユーザー名とパスワードを入力するのはかなり面倒）、かなり使いにくいアプリになります。

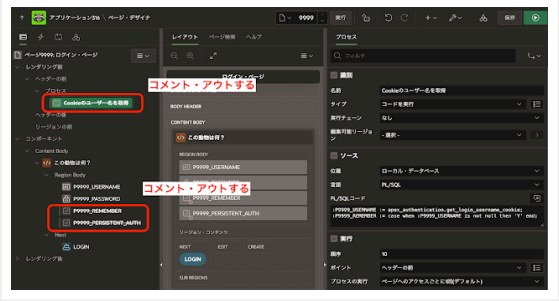
Oracle APEXでは、パスワード入力を一定期間省略するために、永続認証という機能を提供していますが、インスタンス単位で有効にする必要があります。しかし、インスタンスの設定変更はできない場合があります。

アプリケーションのログイン・ページにあるプロセス**Login**の引数**p_set_persistent_auth**の値を常に**True**にすることによって、アプリケーション単位で永続認証を有効にできるようです。

プロセス**Cookieのユーザー名を設定**は**コメント・アウト**します。



レンダリング・ビューのプロセス**Cookieのユーザー名を取得**、ページ・アイテム**P9999_REMEMBER**、**P9999_PERSISTENT_AUTH**もコメント・アウトします。



以上で、サインインする際に永続認証を強制することができます。

今回の記事は以上になります。

Oracle APEXのアプリケーション作成の参考になれば幸いです。

完

Yuji N. 時刻: 11:05

共有

‹ ホーム ›

ウェブ バージョンを表示

自己紹介

Yuji N.

日本オラクル株式会社に勤務していて、Oracle APEXのGroundbreaker Advocateを拝命しました。
こちらの記事につきましては、免責事項の参照をお願いいたします。

詳細プロフィールを表示