日日是Oracle APEX

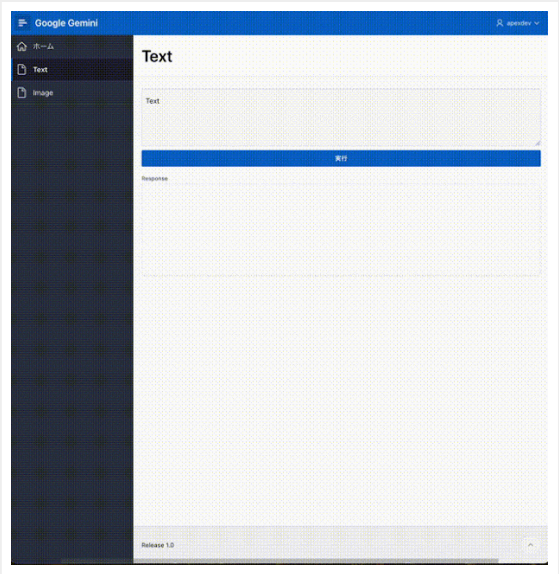
Oracle APEXを使った作業をしていて、気の付いたところを忘れないようにメモをとります。

2023年12月23日土曜日

Google Geminiを呼び出すAPEXアプリケーションを作る

Google AI Gemini APIを呼び出すAPEXアプリケーションを作ってみました。

以下のように動作します。Gemini APIはチャット形式のやり取り（OpenAIのChat Completionsとほぼ同じ）と関数呼び出し（OpenAIのFunction Callingとほぼ同じ）もサポートしていますが、これらの実装はまた別の機会にしようと思います。

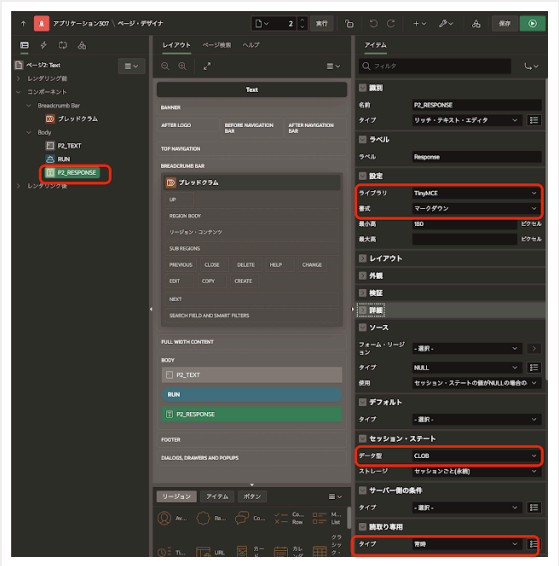


写っている動物の質問に使った画像は以下です。

タヌキなんですがハクビシンといわれています。世界的にみると珍しい動物らしいので、学習されていないのでしょうか。

APEXのアプリケーションの作りは単純です。

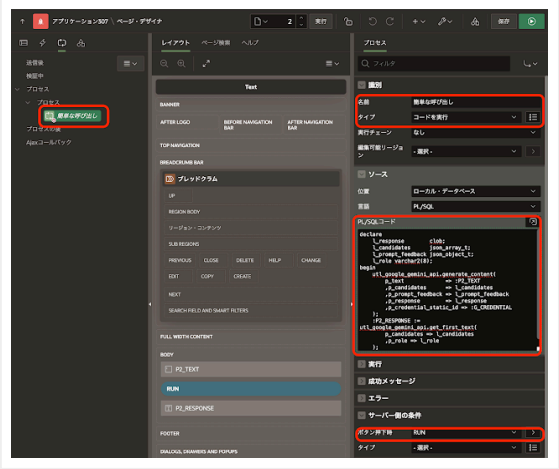
Textのページは、質問を入力する**テキスト領域**のページ・アイテムである**P2_TEXT**、質問を送信ボタン**RUN**、Geminiのレスポンスの**マークダウン**を**読み取り専用**で表示する**P2_RESPONSE**から構成されています。



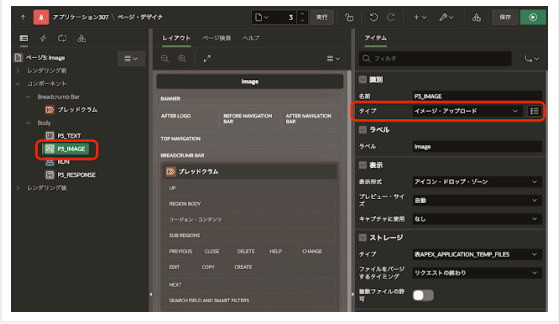
ボタンRUNを押した時に実行されるプロセスとして、以下のコードを記述します。ほとんどの処理はパッケージ**UTL_GOOGLE_GEMINI_API**で行っています。このコードは記事の末尾に添付しています。

```
declare
    l_response        clob;
    l_candidates      json_array_t;
    l_prompt_feedback json_object_t;
    l_role varchar2(8);
begin
    utl_google_gemini_api.generate_content(
        p_text             => :P2_TEXT
        ,p_candidates      => l_candidates
        ,p_prompt_feedback => l_prompt_feedback
        ,p_response        => l_response
        ,p_credential_static_id => :G_CREDENTIAL
    );
    :P2_RESPONSE := utl_google_gemini_api.get_first_text(
        p_candidates => l_candidates
        ,p_role => l_role
    );
end;
```

google-gemini-text.sql hosted with ❤️ by GitHub view raw



画像の問い合わせをするページには、APEX 23.2で新設されたイメージ・アップロードを使っています。

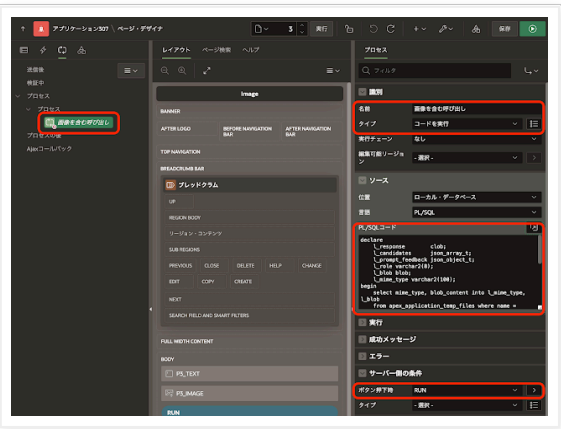


Geminiを呼び出すプロセスのコードは以下になります。

```
declare
    l_response        clob;
    l_candidates      json_array_t;
    l_prompt_feedback json_object_t;
    l_role varchar2(8);
    l_blob blob;
    l_mime_type varchar2(100);
begin
    select mime_type, blob_content into l_mime_type, l_blob
    from apex_application_temp_files where name = :P3_IMAGE;
    utl_google_gemini_api.generate_content(
        p_text             => :P3_TEXT
        ,p_image           => l_blob
        ,p_mimetype        => l_mime_type
        ,p_candidates      => l_candidates
        ,p_prompt_feedback => l_prompt_feedback
        ,p_response        => l_response
        ,p_credential_static_id => :G_CREDENTIAL
    );
    :P3_RESPONSE := utl_google_gemini_api.get_first_text(
        p_candidates => l_candidates
```

```
        ,p_role => l_role
    );
end;
```

google-gemini-pro-vision.sql hosted with ❤️ by GitHub view raw



今回作成したアプリケーションのエクスポートを以下に置きました。
https://github.com/ujnak/apexapps/blob/master/exports/google-gemini.zip

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

```
create or replace package utl_google_gemini_api
as
/*
 * API Reference
 * https://cloud.google.com/vertex-ai/docs/generative-ai/model-reference/gemini
 */
/* threshold */
C_THRESHOLD_BLOCK_NONE             constant varchar2(30) := 'BLOCK_NONE';
C_THRESHOLD_BLOCK_LOW_AND_ABOVE    constant varchar2(30) := 'BLOCK_LOW_AND_ABOVE';
/* BLOCK_MEDIUM_AND_ABOVE or BLOCK_MED_AND_ABOVE ? */
C_THRESHOLD_BLOCK_MEDIUM_AND_ABOVE constant varchar2(30) := 'BLOCK_MEDIUM_AND_ABOVE';
C_THRESHOLD_BLOCK_HIGH_AND_ABOVE   constant varchar2(30) := 'BLOCK_LOW_AND_ABOVE';
/* finishReason */
C_FINISH_REASON_UNSPECIFIED constant varchar2(12) := 'UNSPECIFIED';
C_FINISH_REASON_STOP        constant varchar2(12) := 'STOP';
C_FINISH_REASON_MAX_TOKENS  constant varchar2(12) := 'MAX_TOKENS';
C_FINISH_REASON_SAFETY      constant varchar2(12) := 'SAFETY';
C_FINISH_REASON_RECITATION  constant varchar2(12) := 'RECITATION';
C_FINISH_REASON_OTHER       constant varchar2(12) := 'OTHER';
/* probability */
C_HARM_PROBABILITY_UNSPECIFIED constant varchar2(30) := 'HARM_PROBABILITY_UNSPECIFIED';
C_HARM_PROBABILITY_NEGLIGIBLE  constant varchar2(30) := 'NEGLIGIBLE';
C_HARM_PROBABILITY_LOW         constant varchar2(30) := 'LOW';
C_HARM_PROBABILITY_MEDIUM      constant varchar2(30) := 'MEDIUM';
C_HARM_PROBABILITY_HIGH        constant varchar2(30) := 'HIGH';
```

```
/* API Endpoints gemini-pro or gemini-pro-vision */
C_URL_GENERATE_CONTENT constant varchar2(100) := 'https://generativelanguage.googleapis.com/v1b
C_URL_GENERATE_CONTENT_VISION constant varchar2(100) := 'https://generativelanguage.googleapis.
C_URL_COUNT_TOKENS constant varchar2(100) := 'https://generativelanguage.googleapis.com/v1beta/
C_URL_COUNT_TOKENS_VISION constant varchar2(100) := 'https://generativelanguage.googleapis.com/
C_URL_ENBED_CONTENT constant varchar2(100) := 'https://generativelanguage.googleapis.com/v1beta

/**
 * get first object within parts array which is also first object within candidates array.
 */
function get_first_part(
    p_candidates in json_array_t
    ,p_ignore    in boolean default true
    ,p_role      out varchar2
) return json_object_t;

/**
 * get text value from the first part object.
 */
function get_first_text(
    p_candidates in json_array_t
    ,p_ignore    in boolean default true
    ,p_role      out varchar2
) return clob;

/**
 * if part object is functionCall, call the function and return the response as clob.
 */
function call_function(
    p_part in json_object_t
) return clob;

/**
 * Text-only input, single-turn, model is gemini-pro.
 */
procedure generate_content(
    p_text                  in clob
    -- generationConfig
    ,p_temperature          in number default 0.9
    ,p_topK                 in number default 1
    ,p_topP                 in number default 1
    ,p_max_output_tokens    in number default 2048
    ,p_stop_sequences       in clob   default null
    -- safetySettings
    ,p_harm_category_harassment        in varchar2 default C_THRESHOLD_BLOCK_MEDIUM_AND_ABOVE
    ,p_harm_category_hate_speech       in varchar2 default C_THRESHOLD_BLOCK_MEDIUM_AND_ABOVE
    ,p_harm_category_sexually_explicit in varchar2 default C_THRESHOLD_BLOCK_MEDIUM_AND_ABOVE
```

```
    ,p_harm_category_dangerous_content in varchar2 default C_THRESHOLD_BLOCK_MEDIUM_AND_ABOVE
    ,p_credential_static_id in varchar2
    ,p_candidates      out json_array_t
    ,p_prompt_feedback out json_object_t
    ,p_response        out clob
    ,p_transfer_timeout in number default 180
    /*
     * providing to set p_transfer_timeout for apex_web_service is useful
     * because 180sec is too long for some use cases.
     */
);

/**
 * Text-and-image input, single-turn, model is gemini-pro-vision.
 * Currently, multi-turn is not recommended with Text-and-image.
 */
procedure generate_content(
    p_text                 in clob
    ,p_image               in blob
    ,p_mimetype            in varchar2
    -- generationConfig
    ,p_temperature          in number default 0.4
    ,p_topK                 in number default 32
    ,p_topP                 in number default 1
    ,p_max_output_tokens    in number default 4096
    ,p_stop_sequences       in clob   default null
    -- safetySettings
    ,p_harm_category_harassment        in varchar2 default C_THRESHOLD_BLOCK_MEDIUM_AND_ABOVE
    ,p_harm_category_hate_speech       in varchar2 default C_THRESHOLD_BLOCK_MEDIUM_AND_ABOVE
    ,p_harm_category_sexually_explicit in varchar2 default C_THRESHOLD_BLOCK_MEDIUM_AND_ABOVE
    ,p_harm_category_dangerous_content in varchar2 default C_THRESHOLD_BLOCK_MEDIUM_AND_ABOVE
    ,p_credential_static_id in varchar2
    ,p_candidates      out json_array_t
    ,p_prompt_feedback out json_object_t
    ,p_response        out clob
    ,p_transfer_timeout in number default 180
);

/**
 * Text-and-image or movie, fileURI on Object Storage instead of blob.
 * single-turn, model is gemini-pro-vision
 */
procedure generate_content(
    p_text                 in clob
    ,p_file_uri            in varchar2
    ,p_mimetype            in varchar2
    -- generationConfig
```

```
    ,p_temperature          in number default 0.4
    ,p_topK                 in number default 32
    ,p_topP                 in number default 1
    ,p_max_output_tokens    in number default 4096
    ,p_stop_sequences       in clob   default null
    -- safetySettings
    ,p_harm_category_harassment        in varchar2 default C_THRESHOLD_BLOCK_MEDIUM_AND_ABOVE
    ,p_harm_category_hate_speech       in varchar2 default C_THRESHOLD_BLOCK_MEDIUM_AND_ABOVE
    ,p_harm_category_sexually_explicit in varchar2 default C_THRESHOLD_BLOCK_MEDIUM_AND_ABOVE
    ,p_harm_category_dangerous_content in varchar2 default C_THRESHOLD_BLOCK_MEDIUM_AND_ABOVE
    ,p_credential_static_id in varchar2
    ,p_candidates      out json_array_t
    ,p_prompt_feedback out json_object_t
    ,p_response        out clob
    ,p_transfer_timeout in number default 180
);

/**
 * Text-only input, multi-turn, model is gemini-pro.
 */
procedure generate_content(
    p_contents              in clob
    ,p_tools                in clob default null
    -- generationConfig
    ,p_temperature          in number default 0.9
    ,p_topK                 in number default 1
    ,p_topP                 in number default 1
    ,p_max_output_tokens    in number default 2048
    ,p_stop_sequences       in clob   default null
    -- safetySettings
    ,p_harm_category_harassment        in varchar2 default C_THRESHOLD_BLOCK_MEDIUM_AND_ABOVE
    ,p_harm_category_hate_speech       in varchar2 default C_THRESHOLD_BLOCK_MEDIUM_AND_ABOVE
    ,p_harm_category_sexually_explicit in varchar2 default C_THRESHOLD_BLOCK_MEDIUM_AND_ABOVE
    ,p_harm_category_dangerous_content in varchar2 default C_THRESHOLD_BLOCK_MEDIUM_AND_ABOVE
    ,p_credential_static_id in varchar2
    ,p_candidates      out json_array_t
    ,p_prompt_feedback out json_object_t
    ,p_response        out clob
    ,p_transfer_timeout in number default 180
);

/**
 * Cout tokens.
 */
function count_tokens(
    /* specifiy p_text OR p_contents */
    p_text   in clob default null
```

```
    ,p_parts in clob default null /* for image */
    ,p_credential_static_id in varchar2
) return number;

/**
 * Embedding.
 */
function embed_content(
    p_model   in varchar2 default 'models/embedding-001'
    ,p_text   in clob default null
    ,p_parts  in clob default null /* for image */
    ,p_values out clob
    ,p_credential_static_id in varchar2
) return number;

end utl_google_gemini_api;
/

create or replace package body utl_google_gemini_api
as

/**
 * private function for creating generationConfig object.
 */
function generate_generation_config(
    p_temperature        in number
    ,p_topK              in number
    ,p_topP              in number
    ,p_max_output_tokens in number
    ,p_stop_sequences    in clob
) return json_object_t
as
    l_generation_config json_object_t;
begin
    l_generation_config := json_object_t(
        json_object(
            'temperature' value p_temperature
            ,'topK'       value p_topK
            ,'topP'       value p_topP
            ,'maxOutputTokens' value p_max_output_tokens
            ,'stopSequences' value p_stop_sequences
        )
    );
    return l_generation_config;
end generate_generation_config;

/**
```

```
 * privete function for creating safetySettings array.
 */
function generate_sefety_settings(
    p_harm_category_harassment         in varchar2
    ,p_harm_category_hate_speech       in varchar2
    ,p_harm_category_sexually_explicit in varchar2
    ,p_harm_category_dangerous_content in varchar2
) return json_array_t
as
    l_safety_settings json_array_t := json_array_t();
begin
    l_safety_settings.append(json_object_t(
        json_object(
            'category' value 'HARM_CATEGORY_HARASSMENT'
            ,'threshold' value p_harm_category_harassment
        )
    ));
    l_safety_settings.append(json_object_t(
        json_object(
            'category' value 'HARM_CATEGORY_HATE_SPEECH'
            ,'threshold' value p_harm_category_hate_speech
        )
    ));
    l_safety_settings.append(json_object_t(
        json_object(
            'category' value 'HARM_CATEGORY_SEXUALLY_EXPLICIT'
            ,'threshold' value p_harm_category_sexually_explicit
        )
    ));
    l_safety_settings.append(json_object_t(
        json_object(
            'category' value 'HARM_CATEGORY_DANGEROUS_CONTENT'
            ,'threshold' value p_harm_category_dangerous_content
        )
    ));
    return l_safety_settings;
end generate_sefety_settings;

/**
 * get first part object from the response.
 */
function get_first_part(
    p_candidates in json_array_t
    ,p_ignore    in boolean
    ,p_role      out varchar2
) return json_object_t
as
```

```
    l_candidate json_object_t;
    l_content   json_object_t;
    l_parts     json_array_t;
    l_part      json_object_t;
    l_finish_reason varchar2(20);
    e_too_many_candidates exception;
    e_too_many_parts      exception;
begin
    if p_candidates.get_size() > 1 then
        /* never happen because candidateCount is always 1 at this moment. */
        if not p_ignore then
            raise e_too_many_candidates;
        end if;
    end if;
    l_candidate := treat(p_candidates.get(0) as json_object_t);
    /* assumes finishReason is always "STOP" */
    l_finish_reason := l_candidate.get_string('finishReason');
    if l_finish_reason <>  C_FINISH_REASON_STOP then
        /* not sure how to handle, just log */
        apex_debug.info('finishReason = %', l_finish_reason);
        if l_finish_reason = C_FINISH_REASON_SAFETY then
            apex_debug.info('safetyRatings: %s', l_candidate.get_object('safetyRatings').to_clo
        end if;
    end if;
    /*
     * candidate (object in candidates array) contains finishReason, index, safetyRatings
     * in addition to content.
     */
    l_content := l_candidate.get_object('content');
    p_role    := l_content.get_string('role');
    l_parts   := l_content.get_array('parts');
    if l_parts.get_size() > 1 then
        if not p_ignore then
            raise e_too_many_parts;
        end if;
    end if;
    l_part := treat(l_parts.get(0) as json_object_t);
    return l_part;
end get_first_part;

/**
 * get text value and role from part object in the response.
 */
function get_first_text(
    p_candidates in json_array_t
    ,p_ignore    in boolean
    ,p_role      out varchar2
```

```
) return clob
as
    l_part json_object_t;
begin
    l_part := get_first_part(
        p_candidates => p_candidates
        ,p_ignore    => p_ignore
        ,p_role      => p_role
    );
    return l_part.get_string('text');
end get_first_text;

/**
 * call function that is replied in functionCall.
 * all functions passed in tools must be created as stored procedure.
 * Ref:
 * https://cloud.google.com/vertex-ai/docs/generative-ai/multimodal/function-calling
 */
function call_function(
    p_part in json_object_t
) return clob
as
    l_function      json_object_t;
    l_function_name varchar2(200);
    l_function_args clob;
    l_dynamic_sql   varchar2(4000);
    l_function_out  clob;
    l_response      json_object_t := json_object_t();
    l_function_response_json json_object_t := json_object_t();
    l_function_response clob;
begin
    l_function := p_part.get_object('functionCall');
    if l_function is null then
        /* do nothing */
        return null;
    end if;
    l_function_name := l_function.get_string('name');
    l_function_args := l_function.get_object('args').to_clob();
    /* call stored procedure defined in the database dynamically. */
    l_dynamic_sql := 'begin :a := ' || l_function_name || '(:b); end;';
    execute immediate l_dynamic_sql using in out l_function_out, l_function_args;
    l_function_response_json.put('name', l_function_name);
    l_response.put('name', l_function_name);
    l_response.put('content', json_object_t(l_function_out));
    l_function_response_json.put('response', l_response);
    l_function_response := l_function_response_json.to_clob();
    return l_function_response;
```

```
end call_function;

/**
 * Private procedure to send request to Gemini.
 */
procedure generate_content(
    p_endpoint_url          in varchar2
    ,p_contents             in clob
    ,p_tools                in clob default null
    -- generationConfig
    ,p_temperature          in number
    ,p_topK                 in number
    ,p_topP                 in number
    ,p_max_output_tokens    in number
    ,p_stop_sequences       in clob
    -- safetySettings
    ,p_harm_category_harassment        in varchar2
    ,p_harm_category_hate_speech       in varchar2
    ,p_harm_category_sexually_explicit in varchar2
    ,p_harm_category_dangerous_content in varchar2
    ,p_credential_static_id in varchar2
    ,p_candidates      out json_array_t
    ,p_prompt_feedback out json_object_t
    ,p_response        out clob
    ,p_transfer_timeout in number
)
as
    l_request  json_object_t := json_object_t();
    l_safety_settings json_array_t := json_array_t();
    l_request_clob clob;
    l_response_json   json_object_t;
    l_response        clob;
    e_api_call_failed exception;
begin
    -- contents
    l_request.put('contents', json_array_t(p_contents));
    if p_tools is not null then
        l_request.put('tools', json_array_t(p_tools));
    end if;
    -- generationConfig
    l_request.put('generationConfig',
        generate_generation_config(
            p_temperature => p_temperature
            ,p_topK       => p_topK
            ,p_topP       => p_topP
            ,p_max_output_tokens => p_max_output_tokens
            ,p_stop_sequences    => p_stop_sequences
```

```
        )
    );
    -- safetySettings
    l_request.put('safetySettings',
        generate_sefety_settings(
            p_harm_category_harassment         => p_harm_category_harassment
            ,p_harm_category_hate_speech       => p_harm_category_hate_speech
            ,p_harm_category_sexually_explicit => p_harm_category_sexually_explicit
            ,p_harm_category_dangerous_content => p_harm_category_dangerous_content
        )
    );
    l_request_clob := l_request.to_clob();
    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 => p_endpoint_url
        ,p_http_method => 'POST'
        ,p_body => l_request_clob
        ,p_credential_static_id => p_credential_static_id
        ,p_transfer_timeout => p_transfer_timeout
    );
    if apex_web_service.g_status_code <> 200 then
        raise e_api_call_failed;
    end if;
    p_response := l_response;
    l_response_json   := json_object_t(l_response);
    p_candidates      := l_response_json.get_array('candidates');
    p_prompt_feedback := l_response_json.get_object('promptFeedback');
end generate_content;

/**
 * Text-only input, single-trun, gemini-pro.
 */
procedure generate_content(
    p_text                  in clob
    -- generationConfig
    ,p_temperature          in number
    ,p_topK                 in number
    ,p_topP                 in number
    ,p_max_output_tokens    in number
    ,p_stop_sequences       in clob
    -- safetySettings
    ,p_harm_category_harassment        in varchar2
    ,p_harm_category_hate_speech       in varchar2
    ,p_harm_category_sexually_explicit in varchar2
    ,p_harm_category_dangerous_content in varchar2
    ,p_credential_static_id in varchar2
```

```
    ,p_candidates      out json_array_t
    ,p_prompt_feedback out json_object_t
    ,p_response        out clob
    ,p_transfer_timeout in number
)
as
    l_contents json_array_t  := json_array_t();
    l_content  json_object_t := json_object_t();
    l_parts    json_array_t  := json_array_t();
    l_part     json_object_t := json_object_t();
    l_contents_clob clob;
    l_response clob;
begin
    l_part.put('text', p_text);
    l_parts.append(l_part);
    l_content.put('parts', l_parts);
    l_contents.append(l_content);
    l_contents_clob := l_contents.to_clob();
    generate_content(
        p_endpoint_url       => C_URL_GENERATE_CONTENT
        ,p_contents          => l_contents_clob
        ,p_temperature       => p_temperature
        ,p_topK              => p_topK
        ,p_topP              => p_topP
        ,p_max_output_tokens => p_max_output_tokens
        ,p_stop_sequences    => p_stop_sequences
        ,p_harm_category_harassment        => p_harm_category_harassment
        ,p_harm_category_hate_speech       => p_harm_category_hate_speech
        ,p_harm_category_sexually_explicit => p_harm_category_sexually_explicit
        ,p_harm_category_dangerous_content => p_harm_category_dangerous_content
        ,p_credential_static_id => p_credential_static_id
        ,p_candidates      => p_candidates
        ,p_prompt_feedback => p_prompt_feedback
        ,p_response        => p_response
        ,p_transfer_timeout => p_transfer_timeout
    );
end generate_content;

/**
 * Text-and-image, single-turn, gemini-pro-vision.
 */
procedure generate_content(
    p_text                  in clob
    ,p_image                in blob
    ,p_mimetype             in varchar2
    -- generationConfig
    ,p_temperature          in number
```

```
    ,p_topK                 in number
    ,p_topP                 in number
    ,p_max_output_tokens    in number
    ,p_stop_sequences       in clob
    -- safetySettings
    ,p_harm_category_harassment        in varchar2
    ,p_harm_category_hate_speech       in varchar2
    ,p_harm_category_sexually_explicit in varchar2
    ,p_harm_category_dangerous_content in varchar2
    ,p_credential_static_id in varchar2
    ,p_candidates      out json_array_t
    ,p_prompt_feedback out json_object_t
    ,p_response        out clob
    ,p_transfer_timeout in number
)
as
    l_contents json_array_t  := json_array_t();
    l_content  json_object_t := json_object_t();
    l_parts    json_array_t  := json_array_t();
    l_part     json_object_t;
    l_image_clob clob;
    l_inline_data json_object_t;
    l_contents_clob clob;
begin
    l_part := json_object_t();
    l_part.put('text', p_text);
    l_parts.append(l_part);
    if p_image is not null then
        l_part := json_object_t();
        l_inline_data := json_object_t();
        l_inline_data.put('mimeType', p_mimetype);
        l_image_clob := apex_web_service.blob2clobbase64(p_image, 'N','N');
        l_inline_data.put('data', l_image_clob);
        l_part.put('inlineData', l_inline_data);
        l_parts.append(l_part);
    end if;
    l_content.put('parts', l_parts);
    l_contents.append(l_content);
    l_contents_clob := l_contents.to_clob();
    generate_content(
        p_endpoint_url       => C_URL_GENERATE_CONTENT_VISION
        ,p_contents          => l_contents_clob
        ,p_temperature       => p_temperature
        ,p_topK              => p_topK
        ,p_topP              => p_topP
        ,p_max_output_tokens => p_max_output_tokens
        ,p_stop_sequences    => p_stop_sequences
```

```
        ,p_harm_category_harassment        => p_harm_category_harassment
        ,p_harm_category_hate_speech       => p_harm_category_hate_speech
        ,p_harm_category_sexually_explicit => p_harm_category_sexually_explicit
        ,p_harm_category_dangerous_content => p_harm_category_dangerous_content
        ,p_credential_static_id => p_credential_static_id
        ,p_candidates      => p_candidates
        ,p_prompt_feedback => p_prompt_feedback
        ,p_response        => p_response
        ,p_transfer_timeout => p_transfer_timeout
    );
end generate_content;

/**
 * Text-and-image or movie, single-turn, gemini-pro-vision.
 */
procedure generate_content(
    p_text                  in clob
    ,p_file_uri             in varchar2
    ,p_mimetype             in varchar2
    -- generationConfig
    ,p_temperature          in number
    ,p_topK                 in number
    ,p_topP                 in number
    ,p_max_output_tokens    in number
    ,p_stop_sequences       in clob
    -- safetySettings
    ,p_harm_category_harassment        in varchar2
    ,p_harm_category_hate_speech       in varchar2
    ,p_harm_category_sexually_explicit in varchar2
    ,p_harm_category_dangerous_content in varchar2
    ,p_credential_static_id in varchar2
    ,p_candidates      out json_array_t
    ,p_prompt_feedback out json_object_t
    ,p_response        out clob
    ,p_transfer_timeout in number
)
as
    l_contents json_array_t  := json_array_t();
    l_content  json_object_t := json_object_t();
    l_parts    json_array_t  := json_array_t();
    l_part     json_object_t;
    l_image_clob clob;
    l_file_data json_object_t;
    l_contents_clob clob;
begin
    l_part := json_object_t();
    l_part.put('text', p_text);
```

```
    l_parts.append(l_part);
    if p_file_uri is not null then
        l_part := json_object_t();
        l_file_data := json_object_t();
        l_file_data.put('mimeType', p_mimetype);
        l_file_data.put('fileUri', p_file_uri);
        l_part.put('fileData', l_file_data);
        l_parts.append(l_part);
    end if;
    -- l_content.put('role','user');
    l_content.put('parts', l_parts);
    l_contents.append(l_content);
    l_contents_clob := l_contents.to_clob();
    apex_debug.info(l_contents_clob);
    generate_content(
        p_endpoint_url       => C_URL_GENERATE_CONTENT_VISION
        ,p_contents          => l_contents_clob
        ,p_temperature       => p_temperature
        ,p_topK              => p_topK
        ,p_topP              => p_topP
        ,p_max_output_tokens => p_max_output_tokens
        ,p_stop_sequences    => p_stop_sequences
        ,p_harm_category_harassment        => p_harm_category_harassment
        ,p_harm_category_hate_speech       => p_harm_category_hate_speech
        ,p_harm_category_sexually_explicit => p_harm_category_sexually_explicit
        ,p_harm_category_dangerous_content => p_harm_category_dangerous_content
        ,p_credential_static_id => p_credential_static_id
        ,p_candidates      => p_candidates
        ,p_prompt_feedback => p_prompt_feedback
        ,p_response        => p_response
        ,p_transfer_timeout => p_transfer_timeout
    );
end generate_content;

/**
 * Text-only-input, multi-turn, gemini-pro.
 */
procedure generate_content(
    p_contents              in clob
    ,p_tools                in clob
    -- generationConfig
    ,p_temperature          in number
    ,p_topK                 in number
    ,p_topP                 in number
    ,p_max_output_tokens    in number
    ,p_stop_sequences       in clob
    -- safetySettings
```

```
    ,p_harm_category_harassment        in varchar2
    ,p_harm_category_hate_speech       in varchar2
    ,p_harm_category_sexually_explicit in varchar2
    ,p_harm_category_dangerous_content in varchar2
    ,p_credential_static_id in varchar2
    ,p_candidates      out json_array_t
    ,p_prompt_feedback out json_object_t
    ,p_response        out clob
    ,p_transfer_timeout in number
)
as
    l_contents json_array_t  := json_array_t();
    l_contents_clob clob;
    l_response clob;
begin
    generate_content(
        p_endpoint_url       => C_URL_GENERATE_CONTENT
        ,p_contents          => p_contents
        ,p_tools             => p_tools
        ,p_temperature       => p_temperature
        ,p_topK              => p_topK
        ,p_topP              => p_topP
        ,p_max_output_tokens => p_max_output_tokens
        ,p_stop_sequences    => p_stop_sequences
        ,p_harm_category_harassment        => p_harm_category_harassment
        ,p_harm_category_hate_speech       => p_harm_category_hate_speech
        ,p_harm_category_sexually_explicit => p_harm_category_sexually_explicit
        ,p_harm_category_dangerous_content => p_harm_category_dangerous_content
        ,p_credential_static_id => p_credential_static_id
        ,p_candidates      => p_candidates
        ,p_prompt_feedback => p_prompt_feedback
        ,p_response        => p_response
        ,p_transfer_timeout => p_transfer_timeout
    );
end generate_content;

/**
 * count tokens
 */
function count_tokens(
    p_text   in clob default null
    ,p_parts in clob default null
    ,p_credential_static_id in varchar2
) return number
as
    l_url       varchar2(200);
    l_request  json_object_t := json_object_t();
```

```
    l_request_clob clob;
    l_contents json_array_t  := json_array_t();
    l_content  json_object_t := json_object_t();
    l_parts    json_array_t  := json_array_t();
    l_part     json_object_t := json_object_t();
    l_response clob;
    l_response_json json_object_t;
    l_total_tokens number;
    e_no_arguments exception;
    e_api_call_failed exception;
begin
    if p_text is not null then
        l_url := C_URL_COUNT_TOKENS;
        l_part.put('text', p_text);
        l_parts.append(l_part);
    else
        if p_parts is null then
            raise e_no_arguments;
        end if;
        l_url := C_URL_COUNT_TOKENS_VISION;
        l_parts := json_array_t(p_parts);
    end if;
    l_content.put('parts', l_parts);
    l_contents.append(l_content);
    l_request.put('contents', l_contents);
    l_request_clob := l_request.to_clob();
    -- apex_debug.info(l_request_clob);
    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_url
        ,p_http_method => 'POST'
        ,p_body => l_request_clob
        ,p_credential_static_id => p_credential_static_id
    );
    if apex_web_service.g_status_code <> 200 then
        raise e_api_call_failed;
    end if;
    l_response_json := json_object_t(l_response);
    return l_response_json.get_number('totalTokens');
end count_tokens;

/**
 * Embedding
 * models/embedding-001 supports text only, p_parts can not be used.
 */
function embed_content(
```

```
    p_model   in varchar2 default 'models/embedding-001'
    ,p_text   in clob
    ,p_parts  in clob
    ,p_values out clob
    ,p_credential_static_id in varchar2
) return number
as
    l_request  json_object_t := json_object_t();
    l_request_clob clob;
    l_content  json_object_t := json_object_t();
    l_parts    json_array_t  := json_array_t();
    l_part     json_object_t := json_object_t();
    l_response clob;
    l_response_json json_object_t;
    l_embedding  json_object_t;
    l_values   json_array_t;
    e_no_arguments exception;
    e_api_call_failed exception;
begin
    if p_text is not null then
        l_part.put('text', p_text);
        l_parts.append(l_part);
    else
        if p_parts is null then
            raise e_no_arguments;
        end if;
        l_parts := json_array_t(p_parts);
    end if;
    l_content.put('parts', l_parts);
    l_request.put('model', p_model);
    l_request.put('content', l_content);
    l_request_clob := l_request.to_clob();
    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 => C_URL_ENBED_CONTENT
        ,p_http_method => 'POST'
        ,p_body => l_request_clob
        ,p_credential_static_id => p_credential_static_id
    );
    if apex_web_service.g_status_code <> 200 then
        raise e_api_call_failed;
    end if;
    l_response_json := json_object_t(l_response);
    l_embedding := l_response_json.get_object('embedding');
    l_values := l_embedding.get_array('values');
    p_values := l_values.to_clob();
```

```
    return l_values.get_size();
end embed_content;

end utl_google_gemini_api;
/
```

utl_google_gemini_api.sql hosted with ❤️ by GitHub view raw

Yuji N. 時刻: 8:49

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