```
@startuml

title DocumentAsk 機能フローチャート


start

:Page_Init

- HTTP ヘッダーに Bearer {ApiKey}設定

- ServicePointManager 設定;

:ユーザーが[質問する]をクリック;

:初期化

- メッセージクリア

- 回答パネル非表示

- 出力ボタン無効化;


if (質問テキストが空?) then (はい)

  :"質問内容を入力してください。"を表示;

  stop

else (いいえ)

endif


if (RAG 使用チェック ON?) then (はい)

  :BuildRagContextAsync(質問, topN=20, neighbor=1);
```

:最終プロンプト = 指示文 + 抜粋 + 質問;

note right

BuildRagContextAsync 概要

- embeddings API で質問のベクトル生成

- タイトルらしき語を抽出して LIKE フィルタ

- DocumentEmbeddings を読み出し

- Title ごとの ChunkIndex の NULL/重複を補正

- コサイン類似度で並べ替え上位 topN

- 同一 Title 内で±neighbor を連結

- 抜粋文字列を返す

end note

else (いいえ)

:最終プロンプト = 質問のみ;

endif

:ChatCompletionRequestDto 作成

- Model=ChatModel

- Messages=[user: 最終プロンプト];

if (ChatModel が gpt-5 以外 かつ Temperature 設定あり?) then (はい)

:Temperature を設定;

endif

:PostWithRetryAsync("chat/completions", CreateJsonContent(req));

note right

 PostWithRetryAsync 概要

 - 最大 4 回まで試行

 - 成功→即返却

 - 429/5xx/タイムアウト→指数バックオフ(500ms*attempt^2)で再試行

 - それ以外→そのまま返却

end note

if (HTTP 成功?) then (はい)

 :JSON を逆シリアル化→reply 抽出;

 if (reply が空?) then (はい)

  :例外("応答が空");

 else (いいえ)

  :reply を HTML エンコードして表示;

  :Session["ChatAnswer"]=reply;

  :回答パネル表示;

  :Word/PDF 出力ボタンを有効化;

 endif

else (いいえ)

:例外("Chat API エラー");

endif

note right

例外発生時

- lblMessage にエラー表示

- DBLogger.LogError で記録

end note

:finally: オーバーレイ非表示スクリプト実行;

if (ユーザーが出力ボタンを押下?) then (はい)

if ([Word 出力]?) then (Word)

if (Session に回答なし?) then (はい)

:"出力する回答が見つかりません。"表示;

else (いいえ)

:OpenXML で DOCX 作成

- 見出し「ChatGPT の回答」

- 各行を段落で追加;

:HTTP レスポンスに docx を書き出し

- Content-Type

```
    - Content-Disposition
    - CompleteRequest();
  endif
 else (PDF)
  if (Session に回答なし?) then (はい)
   :"出力する回答が見つかりません。"表示;
  else (いいえ)
   :iTextSharp で PDF 作成
    - MSGothic 埋め込み
    - 見出しと各行を追加;
   :HTTP レスポンスに pdf を書き出し
    - Content-Type
    - Content-Disposition
    - CompleteRequest();
  endif
 endif
else (いいえ)
endif


end
@enduml
```

DocumentAsk 機能フローチャート