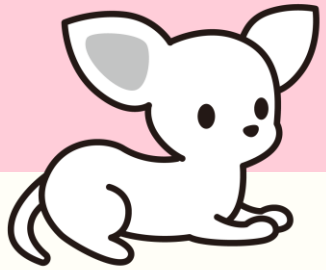
13:00より開始いたします。もうしばらくお待ちください。

スマラジ! SmartDB Radio

活用レベルアップ！

SmartDB深掘りセミナー



スマラジ！複数のAPIを組み合わせてみよう！
～全6回で習得！REST APIアプリ操作応用編③～

株式会社ドリーム・アーツ

協創パートナー推進本部
中川 嵩之

DreamArts

講師

SCS企画・運営担当として奮闘中！

氏名：　中川 嵩之 (なかがわ たかゆき)
所属：　協創パートナー推進本部 CP企画
出身：　埼玉県
経歴：　人事システムベンダーを経て現職
趣味：　ワークアウト

司会

コミュニティで活用を広げたい！

氏名：　山﨑 碧（やまざき みどり）
所属：　協創パートナー推進本部 CP企画
出身：　東京都
経歴：　元プロモーション企業で営業
　　現在はコミュニティ運営担当
趣味：　ラクロス、ゴルフ

ご案内

DreamArts

①チャット手順

画面下にあるメニューに
「チャット」アイコンがあります。
こちらをクリックすると、
チャットのウィンドウが出てきます。

送信先を「全員」にした状態で
チャットしてください。
※宛先が全員でない場合投稿に気づけない場合がございます。

②Q＆A機能利用のお願い

画面下にあるメニューに
「Q＆A」アイコンがあります。
こちらをクリックすると、Q＆Aのウィンドウが
出てきます。

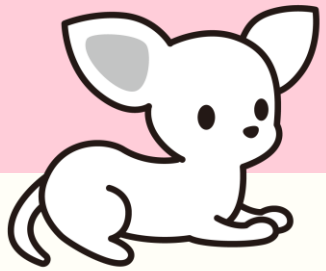
「質問をここに入力してください」の箇所から
質問をしてください。
※質問内容は講師と司会にのみにしか見れないようにしていますのでご安心ください。

- バージョンやご利用方法によって、
  デモでの画面イメージや操作感が一部違う場合がございます。
- 質疑応答に関して、時間の関係上全て回答できない場合があります。
  その場合は、後日コミュニティサイトにて回答します。
- コミュニティサイトへの公開用として録画させていただきます。
- 今後の改善のため、最後にアンケートの回答にご協力ください。

スマラジ！ SmartDB Radio

活用レベルアップ！

SmartDB深掘りセミナー



スマラジ！複数のAPIを組み合わせてみよう！
～全6回で習得！REST APIアプリ操作応用編③～

株式会社ドリーム・アーツ

協創パートナー推進本部
中川 嵩之

DreamArts

1. 前回の復習
2. 応用的なSmartDB REST API
3. まとめ
4. 次回予告

DreamArts

1. 前回の復習
2. 応用的なSmartDB REST API
3. まとめ
4. 次回予告

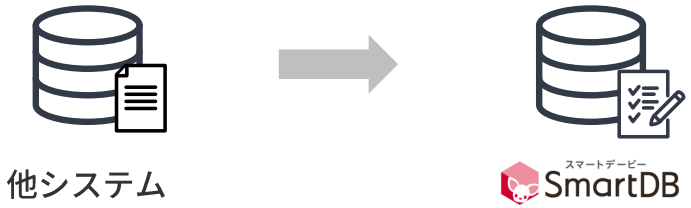
1. SmartDB REST APIを利用するには、バインダトークンが必要
2. SmartDB REST APIを組み合わせることで業務を実現
3. SmartDB REST API（v3）、SmartDB環境情報を参照してリクエスト作成

● 代表的なよく使われるSmartDB REST API一覧

カテゴリ	REST API
バインダ / 文書	文書詳細情報取得 指定したビューの文書一覧取得 文書一覧のCSV出力 文書新規登録 文書更新 文書削除 文書一括操作 CSV入力（新規） CSV入力（更新）
業務プロセス	業務プロセス開始 アクティビティ実施 文書情報に基づいて実行中アクティビティ情報取得

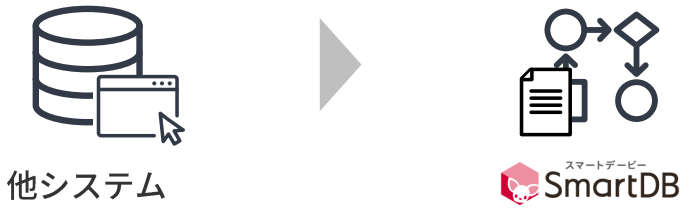
DreamArts

SmartDBの文書を更新



1. 他システムのAPI（GET）
2. 指定したビューの文書一覧取得
3. 文書更新

SmartDBの業務プロセスを開始



1. 他システムのAPI（GET）
2. 指定したビューの文書一覧取得
3. 業務プロセス開始

前回の復習：SmartDB REST APIのリクエスト作成

要素	項目	参照情報
URI	プロトコル	SmartDB REST API（v3）TOPページ ※原則として「https://」を利用
	ホスト名	利用しているSmartDB
	リソースパス	SmartDB REST API（v3）各APIのページ
	クエリパラメータ	項目：SmartDB REST API（v3）各APIのページ 値：利用しているSmartDB
HTTPメソッド	-	SmartDB REST API（v3）各APIのページ
リクエストヘッダ	認証情報	SmartDB REST API（v3）TOPページ ※バインダトークンの場合、「Authorization : Bearer [Token]」を利用
	その他	必要に応じてWeb/ChatGPTで検索 ※SmartDB REST API（v3）「添付ファイル情報取得」には一部記載あり
リクエストボディ	-	項目：SmartDB REST API（v3）各APIのページ 値：利用しているSmartDB

要素	項目	参照情報
リクエストヘッダ	認証情報	SmartDB REST API（v3）TOPページ ※バインダトークンの場合、「Authorization : Bearer [Token]」を利用
	その他	必要に応じてWeb/ChatGPTで検索 ※SmartDB REST API（v3）「添付ファイル情報取得」には一部
リクエストボディ	-	項目：SmartDB REST API（v3）各APIのページ 値：利用しているSmartDB

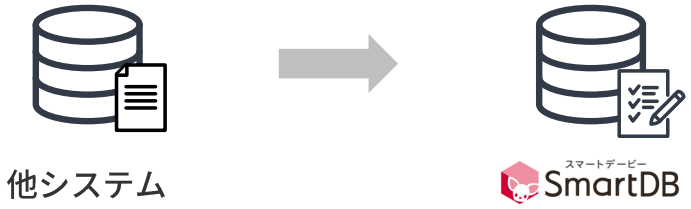
1. SmartDB REST APIで共通する要素は、**SmartDB REST API（v3）TOPページ**を参照
2. SmartDB REST APIごとに異なる要素は、 **SmartDB REST API（v3）各APIのページ**を参照
3. SmartDBの環境ごとに異なる要素/項目/値は、**利用しているSmartDB環境**を参照

DreamArts

1. 前回の復習
2. 応用的なSmartDB REST API
3. まとめ
4. 次回予告

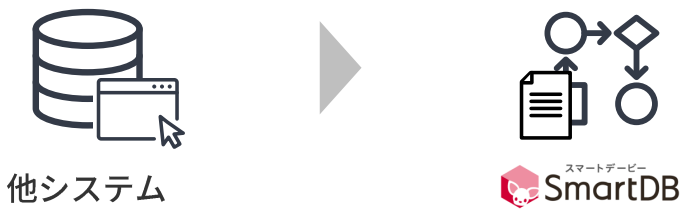
DreamArts

SmartDBの文書を更新



1. 他システムのAPI（GET）
2. 指定したビューの文書一覧取得
3. 文書更新

SmartDBの業務プロセスを開始



1. 他システムのAPI（GET）
2. 指定したビューの文書一覧取得
3. 業務プロセス開始

・
・
・

DreamArts

SmartDBの文書を更新

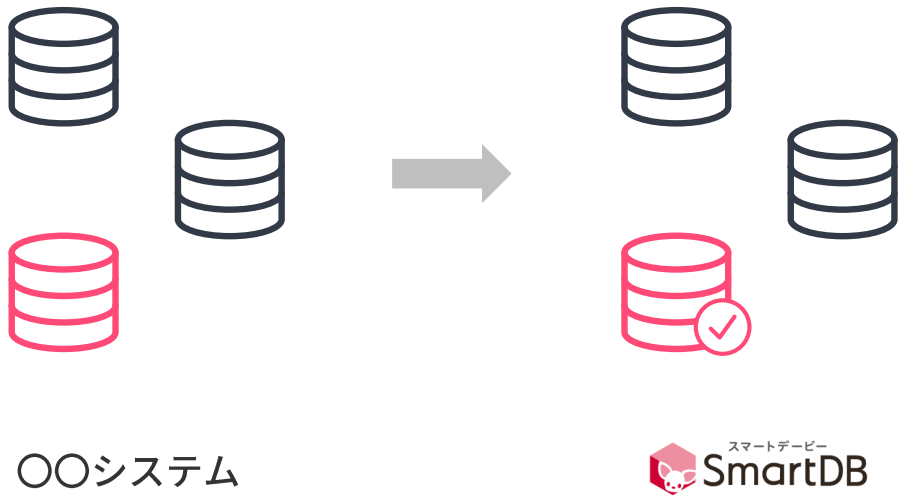
1. 他システムのAPI（GET）
2. 指定したビューの文書一覧取得

やりたいことに応じて大体**決まったAPIの組み合わせ**が存在

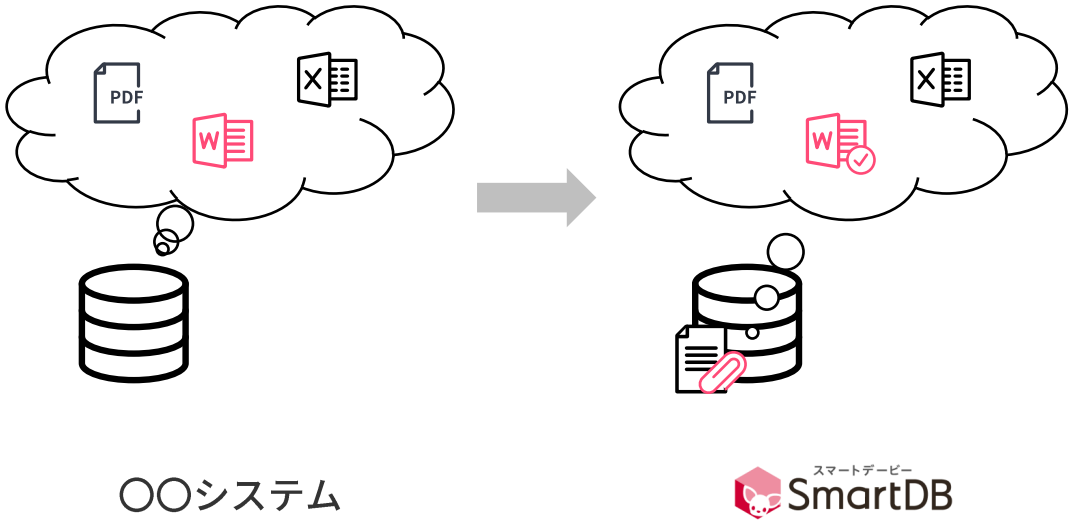
1. 他システムのAPI（GET）
2. 指定したビューの文書一覧取得
3. 業務プロセス開始

DreamArts

マスタバインダの同期



添付ファイルの連携

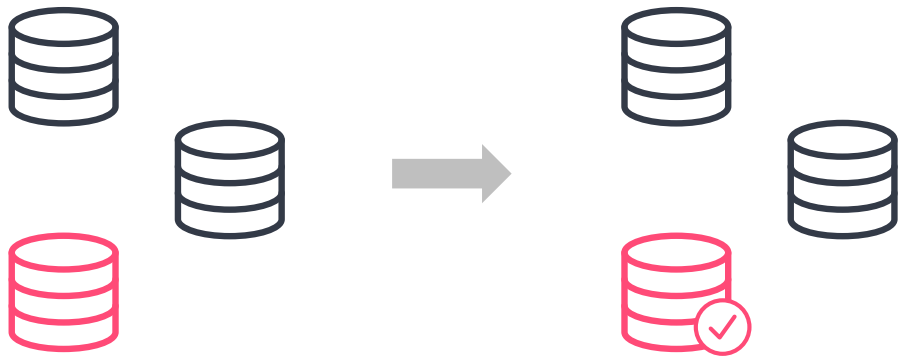


DreamArts

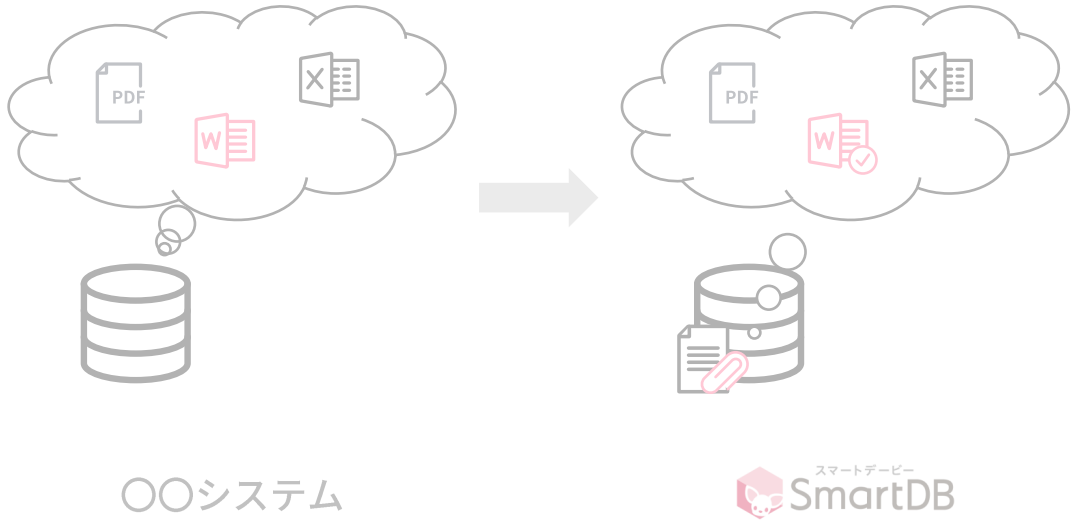
1. 前回の復習
2. 応用的なSmartDB REST API
   - マスタバインダの同期
   - 添付ファイルのアップロード
3. まとめ
4. 次回予告

DreamArts

マスタバインダの同期



添付ファイルの連携



DreamArts

UPSERT

登録更新

新規追加データ、更新されたデータを連携して同期

REPLACE

全量入替

全てのデータを連携して同期

DreamArts

新規追加データ、更新されたデータを連携して同期

他システム

No	会社ID	会社名	所在地	…
1	A0001	株式会社ドリーム・アーツ	東京都渋谷区	
2	A0002	株式会社夢芸	東京都渋谷区	
3	B0001	夢創信息（大連）有限公司	大連市	
		⋮		

SmartDB

No	会社ID	会社名	所在地	…
1	A0001	株式会社ドリーム・アーツ	東京都新宿	
2	A0002	株式会社夢芸	東京都渋谷区	
		⋮		

既存データの更新

No	会社ID	会社名	所在地	…
1	A0001	株式会社ドリーム・アーツ	東京都渋谷区	
2	A0002	株式会社夢芸	東京都渋谷区	
3	B0001	夢創信息（大連）有限公司	大連市	
		⋮		

新規データの追加

No	会社ID	会社名	所在地	…
1	A0001	株式会社ドリーム・アーツ	東京都渋谷区	
2	A0002	株式会社夢芸	東京都渋谷区	
3	B0001	夢創信息（大連）有限公司	大連市	
		⋮		

※既存データの削除は実施されません。

DreamArts

新規追加データ、更新されたデータを連携して同期

他システム

No	会社ID	会社名	所在地	…
1	A0001	株式会社ドリーム・アーツ	東京都渋谷区	
2	A0002	株式会社夢芸	東京都渋谷区	
3	B0001	夢創信息（大連）有限公司	大連市	
		⋮		

SmartDB

No	会社ID	会社名	所在地	…
1	A0001	株式会社ドリーム・アーツ	東京都新宿	
2	A0002	株式会社夢芸	東京都渋谷区	
		⋮		

既存データの更新

No	会社ID	会社名	所在地	…
1	A0001	株式会社ドリーム・アーツ	東京都渋谷区	
2	A0002	株式会社夢芸	東京都渋谷区	
3	B0001	夢創信息（大連）有限公司	大連市	
		⋮		

新規データの追加

No	会社ID	会社名	所在地	…
1	A0001	株式会社ドリーム・アーツ	東京都渋谷区	
2	A0002	株式会社夢芸	東京都渋谷区	
3	B0001	夢創信息（大連）有限公司	大連市	
		⋮		

※既存データの削除は実施されません。

DreamArts

全てのデータを連携して同期

他システム

No	会社ID	会社名	所在地	…
1	A0001	株式会社ドリーム・アーツ	東京都渋谷区	
2	B0001	夢創信息（大連）有限公司	大連市	
		⋮		

SmartDB

No	会社ID	会社名	所在地	…
1	A0001	株式会社ドリーム・アーツ	東京都新宿	
2	A0002	株式会社夢芸	東京都渋谷区	
		⋮		

既存データの更新

No	会社ID	会社名	所在地	…
1	A0001	株式会社ドリーム・アーツ	東京都渋谷区	
2	B0001	夢創信息（大連）有限公司	大連市	
		⋮		

新規データの追加

No	会社ID	会社名	所在地	…
1	A0001	株式会社ドリーム・アーツ	東京都渋谷区	
2	B0001	夢創信息（大連）有限公司	既存データの削除	
		⋮		

※連携されないデータは削除されます。

DreamArts

全てのデータを連携して同期

他システム

No	会社ID	会社名	所在地	…
1	A0001	株式会社ドリーム・アーツ	東京都渋谷区	
2	B0001	夢創信息（大連）有限公司	大連市	
		⋮		

SmartDB

No	会社ID	会社名	所在地	…
1	A0001	株式会社ドリーム・アーツ	東京都新宿	
2	A0002	株式会社夢芸	東京都渋谷区	
		⋮		

既存データの更新

No	会社ID	会社名	所在地	…
1	A0001	株式会社ドリーム・アーツ	東京都渋谷区	
2	B0001	夢創信息（大連）有限公司	大連市	
		⋮		

新規データの追加

No	会社ID	会社名	所在地	…
1	A0001	株式会社ドリーム・アーツ	東京都渋谷区	
2	B0001	夢創信息（大連）有限公司	既存データの削除	
		⋮		

※連携されないデータは削除されます。

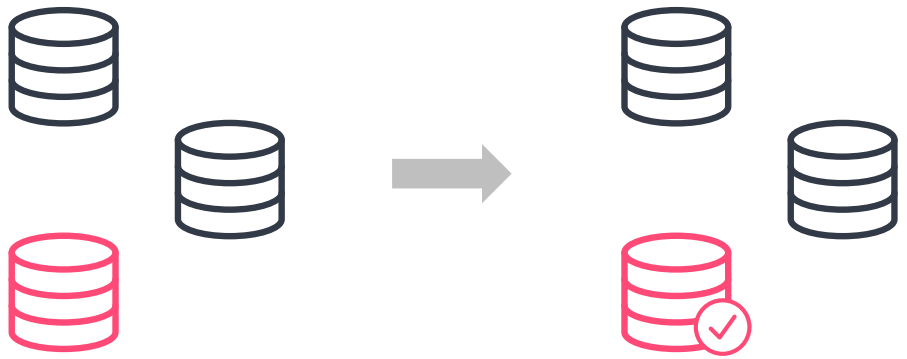
DreamArts

UPSERT REPLACE

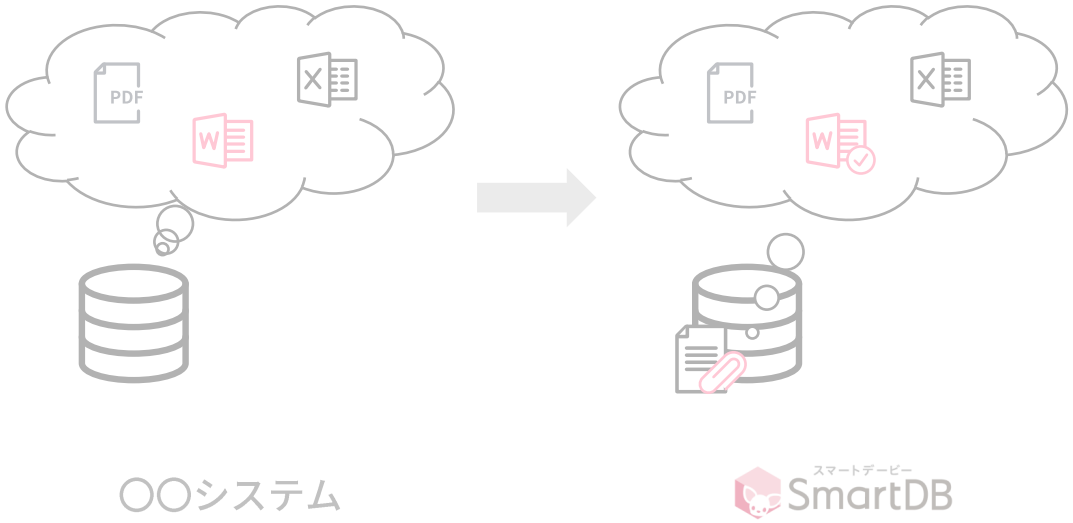
文書の登録、更新（、削除）を1回の処理で実施することができる

DreamArts

マスタバインダの同期



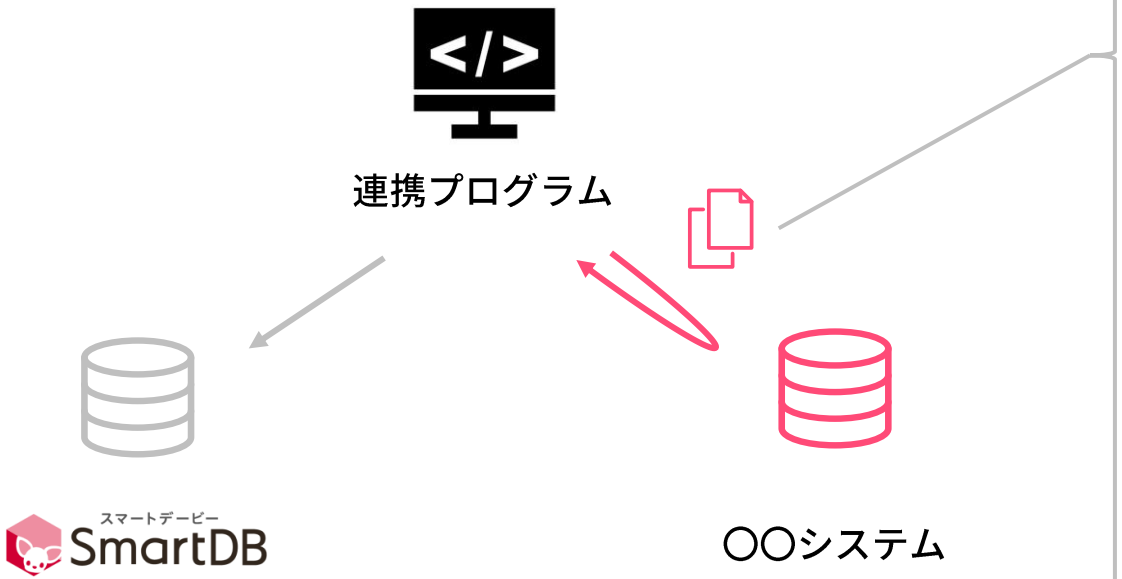
添付ファイルの連携



DreamArts

STEP 01 連携データ取得 → STEP 02 CSV作成 → STEP 03 Base64エンコード → STEP 04 CSV入力

対向システムのAPIでマスタデータを取得



JSON形式のデータ

```
({
   "id": "A0001"
   "name" : "株式会社ドリーム・アーツ",
   "location" : "東京都渋谷区"
   }, {
   "id" : "A0002",
   "name" : "株式会社夢芸",
   "location" : "東京都渋谷区"
})
```

or

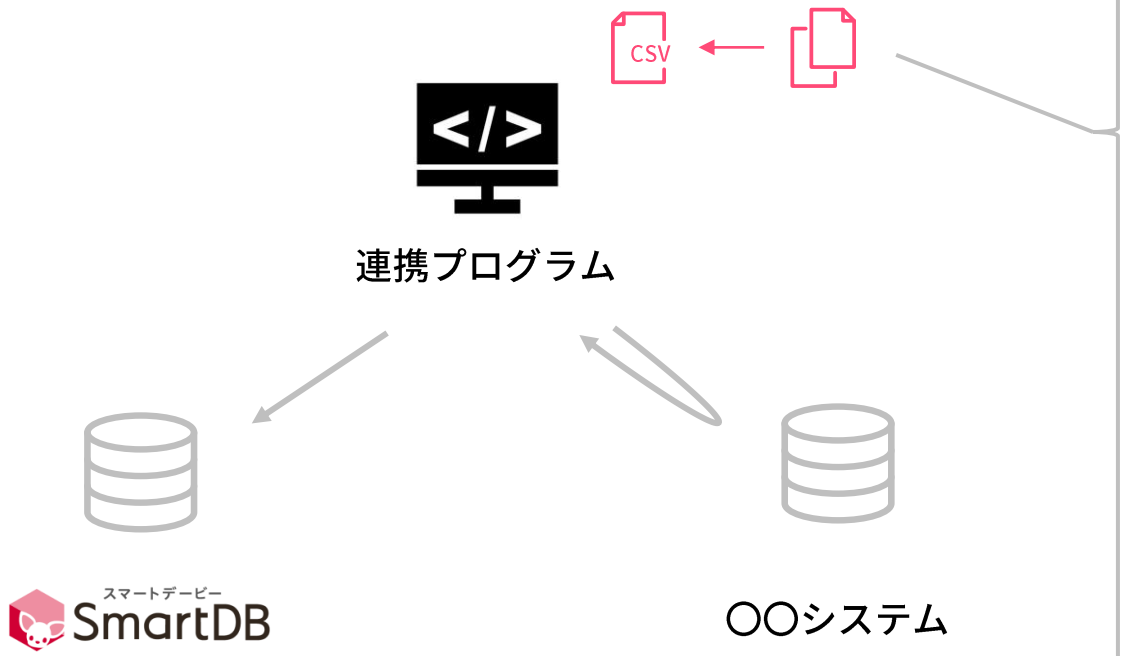
CSV形式のデータ

```
id,name,location
A0001,株式会社ドリーム・アーツ,東京都渋谷区
A0002,株式会社夢芸,東京都渋谷区
```

DreamArts

STEP 01 連携データ取得 → STEP 02 CSV作成 → STEP 03 Base64エンコード → STEP 04 CSV入力

SmartDBの部品キーに項目をマッピングして登録/更新CSVを作成



JSON形式のデータ

```
({
   "id": "A0001"
   "name" : "株式会社ドリーム・アーツ",
   "location" : "東京都渋谷区"
   }, {
   "id" : "A0002",
   "name" : "株式会社夢芸",
   "location" : "東京都渋谷区"
})
```

ヘッダーを「[部品キー]項目名」に変換

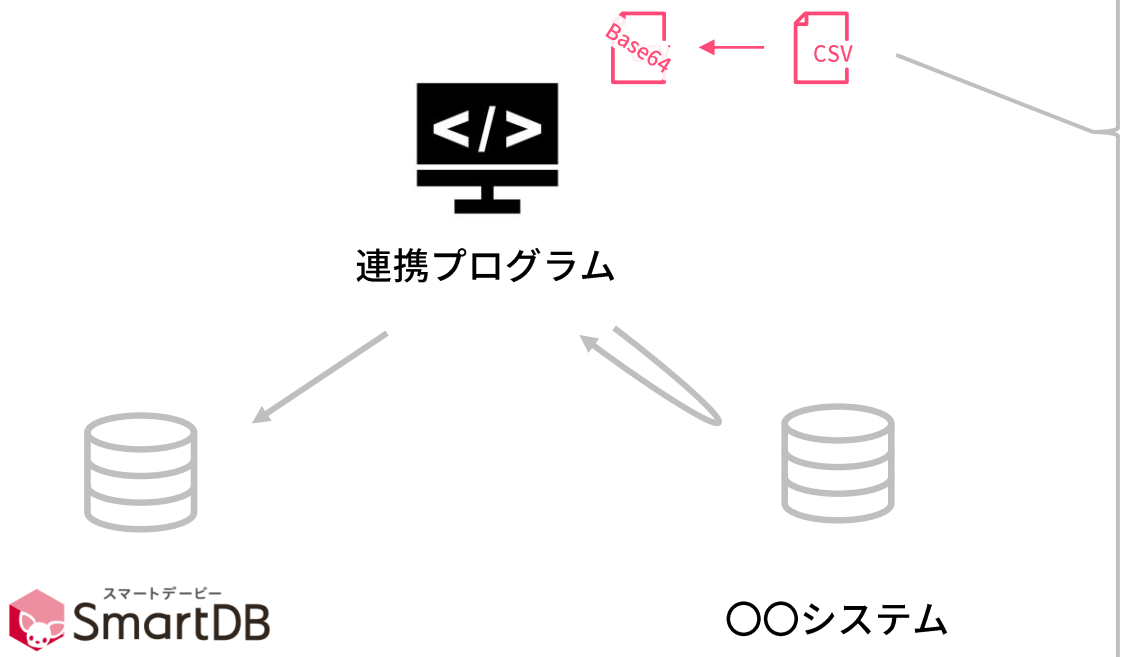
CSV形式のデータ

```
[companyId]会社ID,[companyName]会社名, [companyAdress]所在地
A0001,株式会社ドリーム・アーツ,東京都渋谷区
A0002,株式会社夢芸,東京都渋谷区
```

DreamArts

STEP 01 連携データ取得 → STEP 02 CSV作成 → STEP 03 Base64エンコード → STEP 04 CSV入力

CSVデータをBase64でエンコード



CSV形式のデータ

```
[companyId]会社ID,[companyName]会社名, [companyAdress]所在地
A0001,株式会社ドリーム・アーツ,東京都渋谷区
A0002,株式会社夢芸,東京都渋谷区
```

文字列形式のデータ

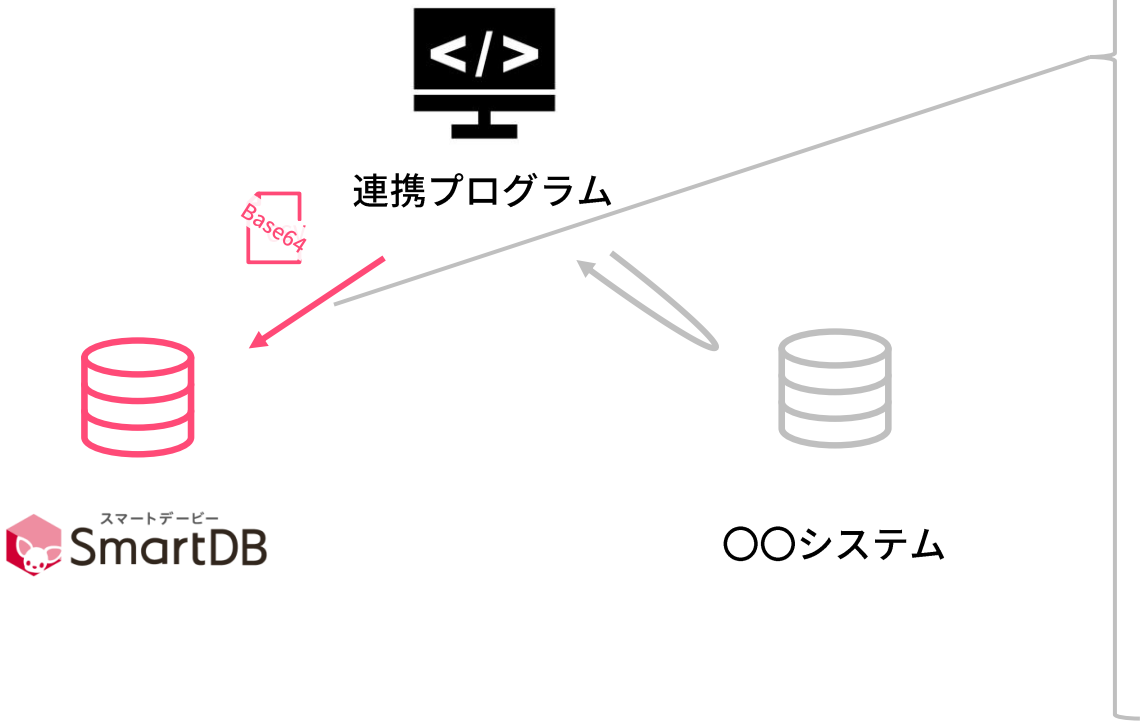
```
IklzUmVhZE9ubHkiLCJJc0ZpeGVkU2l6ZSIsIklzU3luY2hyb25pemVkIiwiS2V5cyIsIlZhbHVlcyIsIlN5bmNSb290IiwiQ291bnQiCiJGYWxzZSIsIkZhbHNlIiwiRmFsc2UiLCJTeXN0ZW0uQ29sbGVjdGlvbnMuSGFzaHRhYmxlK0tleUNvbGxlY
ファイルはREST APIで送信できないため
REST APIで送れる文字列の形式に変換
hc2h0YWJsZStLZXlDb2xsZWN0aW9uIiwiU3lzdGVtLkNvbGxlY3Rpb25zLkhhc2h0YWJsZStWYWx1ZUNvbGxlY3Rpb24iLCJTeXN0ZW0uT2JqZWN0IiwiMyI=
```

マスタバインダの同期手順

DreamArts

STEP 01 連携データ取得 → STEP 02 CSV作成 → STEP 03 Base64エンコード → STEP 04 CSV入力

SmartDB REST API「CSV入力（更新）」を実行



リクエスト

HTTPメソッド / リソースパス

PUT /binders/{binder}/documents/csv

リクエストヘッダ

Authorization：Bearer バインダトークン

Content-Type：application/json

リクエストボディ

マスタ同期では、"UPSERT"か"REPLACE"を指定

csv ：Base64でエンコードしたCSVファイル

updateMode ：更新方法

keyItemId ：キーとして利用する部品

DreamArts

問題 マスタ同期で「CSV入力（更新）」を使う理由として正しいものは？

1. UPSERT、REPLACEを指定することができるため
2. JSON形式のデータと比較してCSV形式の方が扱いやすいため
3. Base64でエンコードしてデータを取り扱うことができるため
4. 特に理由はない

DreamArts

問題　マスタ同期で「CSV入力（更新）」を使う理由として正しいものは？

1. **UPSERT、REPLACEを指定することができるため**
2. JSON形式のデータと比較してCSV形式の方が扱いやすいため
3. Base64でエンコードしてデータを取り扱うことができるため

解説

1. UPSERTにより**登録更新**を、REPLACEにより**全量入替**を行うことができるため、**一回の処理**で同期を完了することができる
2. 今回の場合、扱うデータ形式はAPIの選定理由とは無関係
   ※そもそも、個人的にはJSON形式の方が扱いやすい
3. CSVデータのBase64エンコードは必須事項であり、APIの選定理由とは無関係

DreamArts

問題

「CSV入力（更新）」の説明として正しいものは？

※ SmartDB REST API（v3）「 CSV入力（更新）」を調べながら回答してください

1. HTTPメソッドは"POST"を指定する
2. リクエストボディの"updateMpde"で"UPSERT"を指定した場合、全量入替が行われる
3. リクエストボディの"csv"には、Base64でエンコードしたCSVデータを指定する必要がある
4. 認証情報は特に指定する必要がない

DreamArts

問題

「CSV入力（更新）」の説明として正しいものは？

※ SmartDB REST API（v3）「 CSV入力（更新） 」を調べながら回答してください

1. HTTPメソッドは"POST"を指定する

2. リクエストボディの"updateMpde"で"UPSERT"を指定した場合、全量入替が行われる

3. **リクエストボディの"csv"には、Base64でエンコードしたCSVデータを指定する必要がある**

解説

1. 「CSV入力（更新）」はPUT を指定する
2. UPSERTは登録更新で、REPLACEが全量入替となる
3. "csv"にはBase64エンコードしたCSVデータを指定する必要がある
4. SmartDB REST APIでは基本的に認証を行う必要がある

マスタバインダの同期：PowerShellで実践

DreamArts

他システム

No	会社ID	会社名	所在地
1	A0001	株式会社ドリーム・アーツ	東京都渋谷区
2	A0002	株式会社夢芸	東京都渋谷区
3	B0001	夢創信息（大連）有限公司	大連市

SmartDB

マスタバインダの同期：PowerShellのスクリプト

```
# 0.共通情報
$sdbUri = "https://【sdbDmain】.smartdb.jp/hibiki/rest/3"
$binderToken = "【binderToken】"
$binderId = "【binderId】"
$keyMap = @{
  "companyId" = "[companyId]会社Id"
  "companyName" = "[companyName]会社名"
  "location" = "[companyAddress]所在地"
}

# 1.連携データ取得　※サンプルデータ
$sampleData = @{data = @(
  @{
    "regDate" = "2000/01/01"
    "companyId"= "A0001"
    "companyName" = "株式会社ドリーム・アーツ"
    "location" = "東京都渋谷区"
    "updDate" = "2023/01/01"
  }, @{
    "regDate" = "2000/01/01"
    "companyId"= "B0001"
    "companyName" = "夢創信息（大連）有限公司"
    "location" = "大連市"
    "updDate" = "2023/01/01"
  }
)} | ConvertTo-Json

# 2.データマッピングを実施したCSVの作成
$sampleDataFromJson = $sampleData | ConvertFrom-Json
$companyDataTbl = $sampleDataFromJson.data | ForEach-Object { [PSCustomObject]@{
  $keyMap.companyId = $_.companyId
  $keyMap.companyName = $_.companyName
  $keyMap.location = $_.location
}}

# 3.CSVデータをバイナリデータに変換 ※可読性の観点から複数行に分割
$companyDataCsv = $companyDataTbl | ConvertTo-Csv -NoTypeInformation
$companyDataStr = [system.String]::Join("`n", $companyDataCsv)
$companyDataBytes = [System.Text.Encoding]::UTF8.GetBytes($companyDataStr)
$companyDataBase64Str = [System.Convert]::ToBase64String($companyDataBytes)

# 4.文書データの更新：SmartDB REST API「CSV入力（更新）」
$uri1 = "$sdbUri/binders/$BinderId/documents/csv"
$method1 = "PUT"
$headers1 = @{
          "Authorization" = "Bearer $binderToken"
          "Content-Type" = "application/json"
        }
$body1 = @{
        "csv" = $companyDataBase64Str
        "charset" = "UTF-8"
        "updateMode" = "UPSERT"
        "keyItemId" = "companyId"
      } | ConvertTo-Json
$body1
$body1ToBytes = [System.Text.Encoding]::UTF8.GetBytes($body1)
Invoke-WebRequest $uri1 -Method $method1 -Headers $headers1 -Body $body1 -UseBasicParsing -
OutFile .¥response_UpsertMasterBinder.json
```

マスタバインダの同期：PowerShellのスクリプト

DreamArts

STEP 00　共通情報整理

```
# 0.共通情報
$sdbUri = "https://【sdbDmain】.smartdb.jp/hibiki/rest/3"
$binderToken = "【binderToken】"
$binderId = "【binderId】"
$keyMap = @{
  "companyId" = "[companyId]会社Id"
  "companyName" = "[companyName]会社名"
  "location" = "[companyAddress]所在地"
}
```

共通利用する情報を変数に格納

STEP 01　連携データ取得

```
# 1.連携データ取得　※サンプルデータ
$sampleData = @{data = @(
  @{
    "regDate" = "2000/01/01"
    "companyId"= "A0001"
    "companyName" = "株式会社ドリーム・アーツ"
    "location" = "東京都渋谷区"
    "updDate" = "2023/01/01"
  }, @{
    "regDate" = "2000/01/01"
    "companyId"= "B0001"
    "companyName" = "夢創信息（大連）有限公司"
    "location" = "大連市"
    "updDate" = "2023/01/01"
  }
)} | ConvertTo-Json
```

他システムのAPIでデータ取得

※今回は対向システムがないのでサンプルデータを作成

STEP 02　CSV作成

```
# 2.データマッピングを実施したCSVの作成
$sampleDataFromJson = $sampleData | ConvertFrom-Json
$companyDataTbl = $sampleDataFromJson.data | ForEach-Object { [PSCustomObject]@{
  $keyMap.companyId = $_.companyId
  $keyMap.companyName = $_.companyName
  $keyMap.location = $_.location
}}
```

取得したデータからCSV作成

STEP 03　Base64エンコード

```
# 3.CSVデータをバイナリデータに変換 ※可読性の観点から複数行に分割
$companyDataCsv = $companyDataTbl | ConvertTo-Csv -NoTypeInformation
$companyDataStr = [system.String]::Join("`n", $companyDataCsv)
$companyDataBytes = [System.Text.Encoding]::UTF8.GetBytes($companyDataStr)
$companyDataBase64Str = [System.Convert]::ToBase64String($companyDataBytes)
```

CSVをBase64でエンコーディング

STEP 04　CSV入力

```
# 4.文書データの更新：SmartDB REST API「CSV入力（更新）」
$uri1 = "$sdbUri/binders/$BinderId/documents/csv"
$method1 = "PUT"
$headers1 = @{
        "Authorization" = "Bearer $binderToken"
        "Content-Type" = "application/json"
      }
$body1 = @{
      "csv" = $companyDataBase64Str
      "charset" = "UTF-8"
      "updateMode" = "UPSERT"
      "keyItemId" = "companyId"
    } | ConvertTo-Json
$body1
$body1ToBytes = [System.Text.Encoding]::UTF8.GetBytes($body1)
Invoke-WebRequest $uri1 -Method $method1 -Headers $headers1 -Body $body1 -UseBasicParsing -
OutFile .¥response_UpsertMasterBinder.json
```

SmartDB REST API「文書更新」を実行

マスタバインダの同期：PowerShellのスクリプト

DreamArts

```
# 0.共通情報
$sdbUri = "https://【sdbDmain】.smartdb.jp/hibiki/rest/3"
$binderToken = "【binderToken】"
$binderId = "【binderId】"
$keyMap = @{
  "companyId" = "[companyId]会社Id"
  "companyName" = "[companyName]会社名"
  "location" = "[companyAddress]所在地"
}

# 1.連携データ取得　※サンプルデータ
$sampleData = @{data = @(
  @{
    "regDate" = "2000/01/01"
    "companyId"= '
    "companyNam
    "location" = "東
    "updDate" = "2023/01/01"
  }, @{
    "regDate" = "2000/01/01"
    "companyId"= "B0001"
    "companyName" = "夢創信息（大連）有限公司"
    "location" = "大連市"
    "updDate" = "2023/01/01"
  }
)} | ConvertTo-Json
```

詳細なスクリプトの解説はサポートサイトに掲載予定

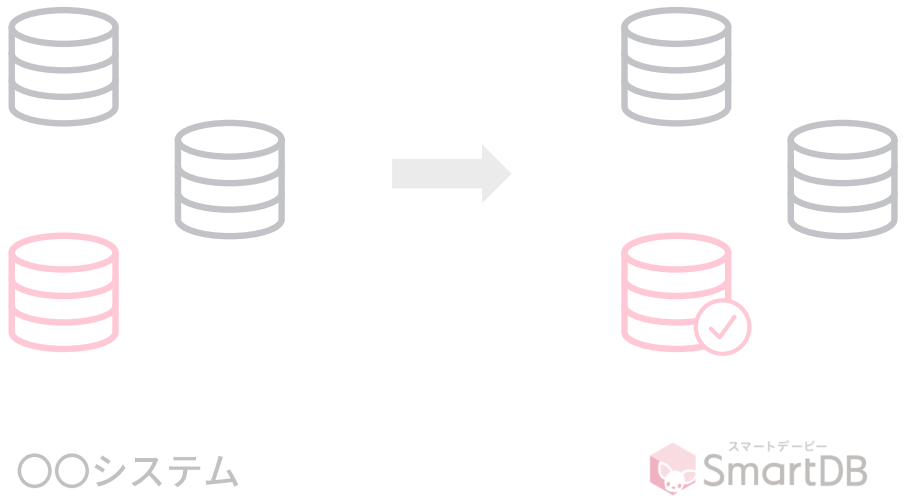
```
# 2.データマッピングを実施したCSVの作成
$sampleDataFromJson = $sampleData | ConvertFrom-Json
$companyDataTbl = $sampleDataFromJson.data | ForEach-Object { [PSCustomObject]@{
  $keyMap.companyId = $_.companyId
  $keyMap.companyName = $_.companyName
  $keyMap.location = $_.location
}}

# 3.CSVデータをバイナリデータに変換 ※可読性の観点から複数行に分割
$companyDataCsv = $companyDataTbl | ConvertTo-Csv -NoTypeInformation
$companyDataStr = [system.String]::Join("`n", $companyDataCsv)
$companyDataBytes = [System.Text.Encoding]::UTF8.GetBytes($companyDataStr)
$companyDataBase64Str = [System.Convert]::ToBase64String($companyDataBytes)

$headers1 = @{
          "Authorization" = "Bearer $binderToken"
          "Content-Type" = "application/json"
        }
$body1 = @{
        "csv" = $companyDataBase64Str
        "charset" = "UTF-8"
        "updateMode" = "UPSERT"
        "keyItemId" = "companyId"
      } | ConvertTo-Json
$body1
$body1ToBytes = [System.Text.Encoding]::UTF8.GetBytes($body1)
Invoke-WebRequest $uri1 -Method $method1 -Headers $headers1 -Body $body1 -UseBasicParsing -
OutFile .¥response_UpsertMasterBinder.json
```

DreamArts

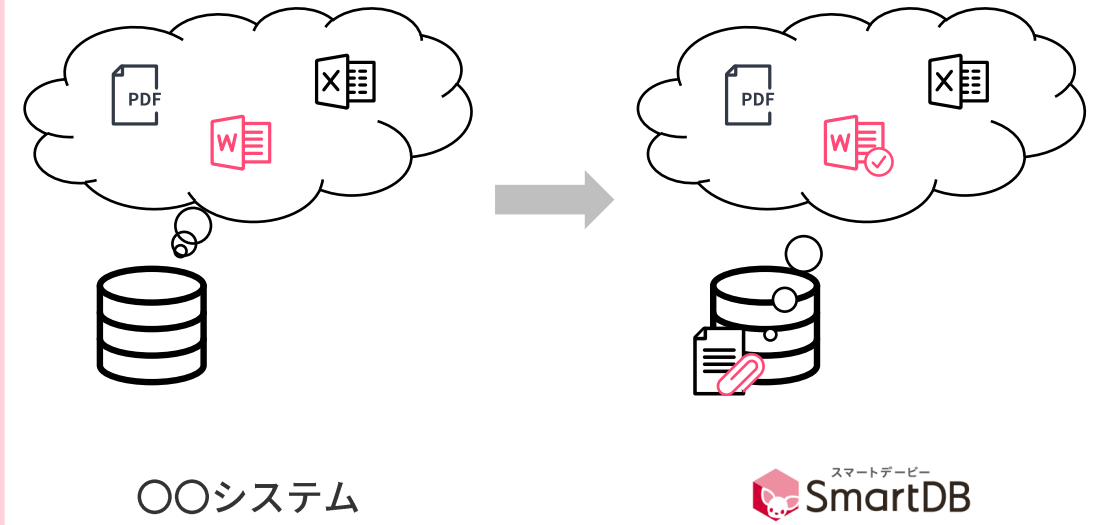
1. 前回の復習
2. 応用的なSmartDB REST API
   - マスタバインダの同期
   - 添付ファイルのアップロード
3. まとめ
4. 次回予告

DreamArts

マスタバインダの同期



添付ファイルの連携



DreamArts

スマートデービー SmartDB

画面で見ている領域

裏側の領域

ここには添付ファイルはいない！

××バインダ

File ID

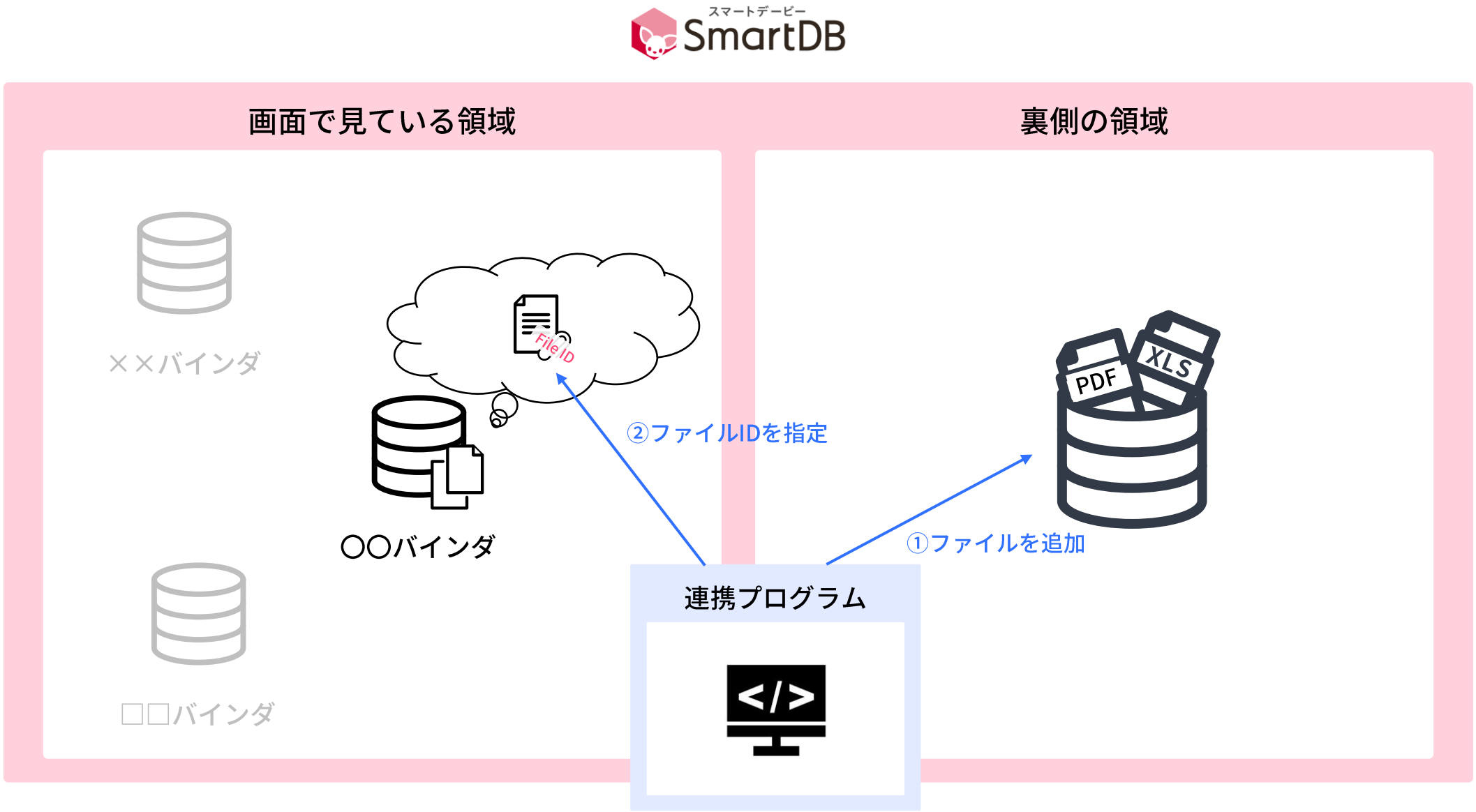
ファイルIDで参照

PDF

XLS

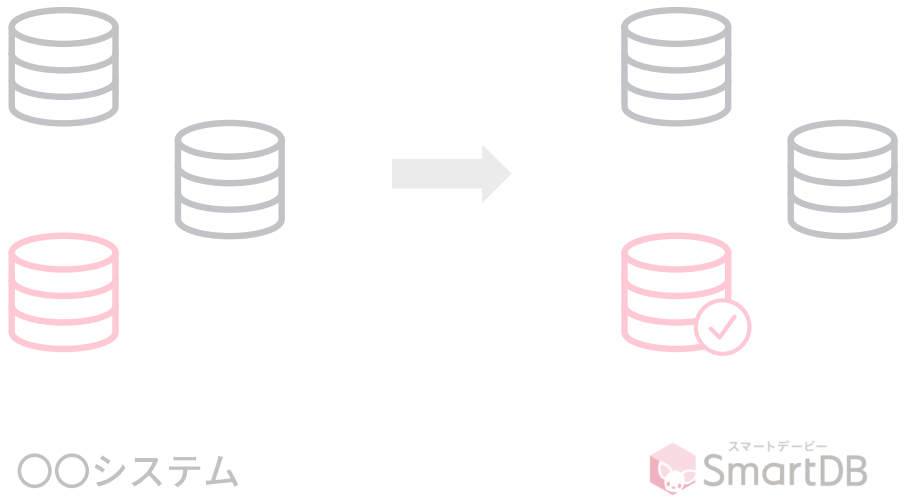
〇〇バインダ

□□バインダ

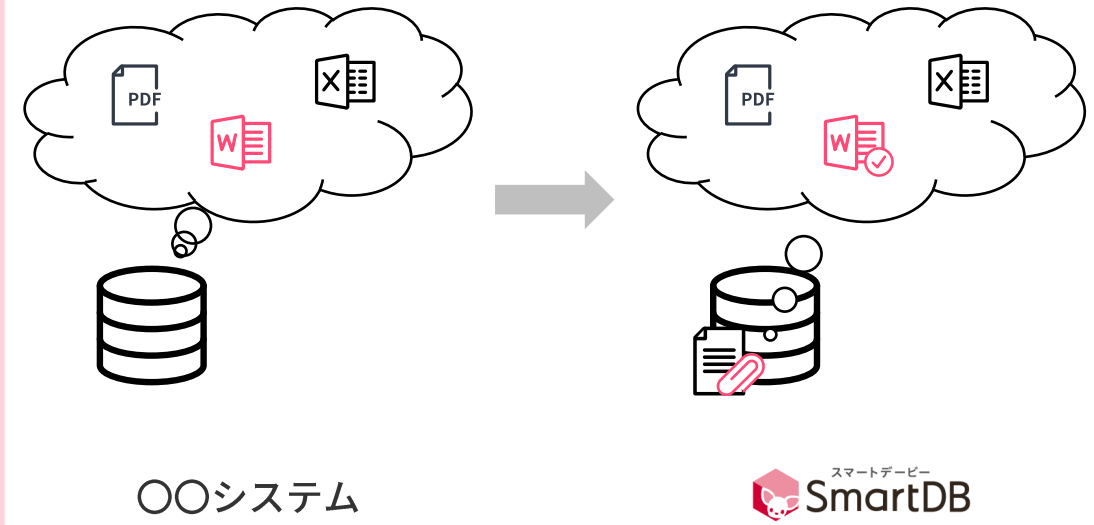


DreamArts

マスタバインダの同期



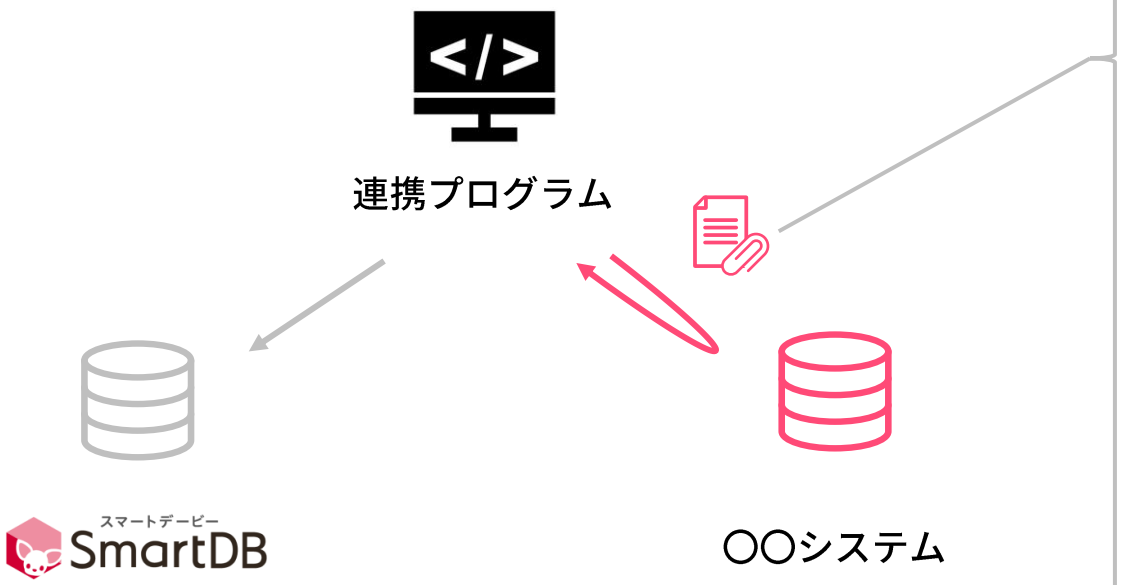
添付ファイルの連携



DreamArts

STEP 01 連携データ取得 | STEP 02 文書情報取得 | STEP 03 新規添付ファイル追加 | STEP 04 ファイルID作成 | STEP 05 文書更新

対向システムのAPIでデータ（ファイルを含む）を取得



JSON形式のデータ

```
{
   "id": "0001"
   "tytle" : "〇〇申請",
   "fileId" : "xxxxxxxxxxxxxxxxxxx"
}
```

＋

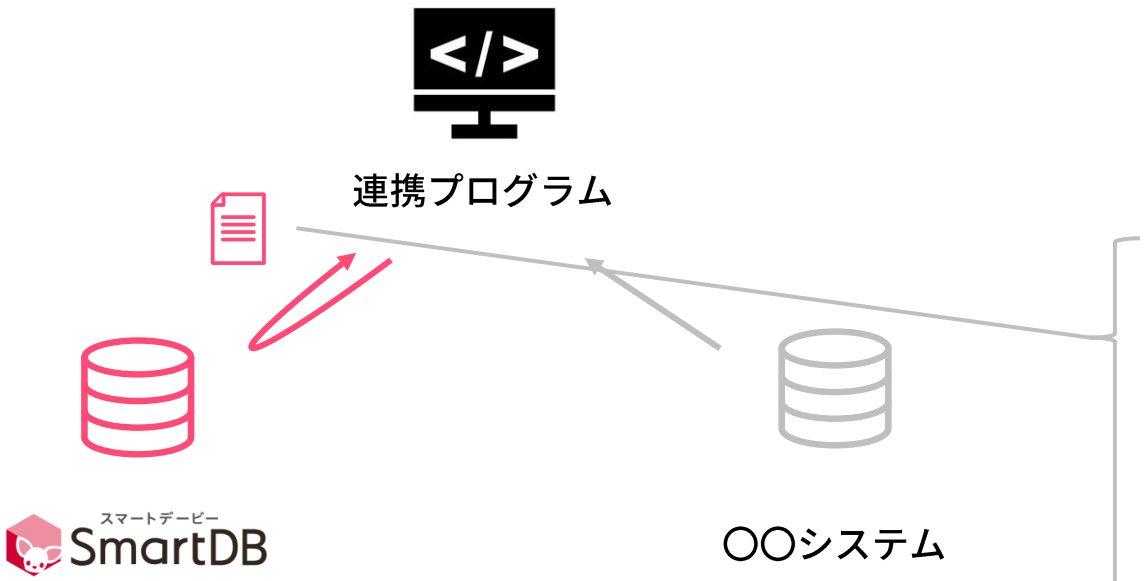
文字列形式のデータ：添付ファイル

```
IklzUmVhZE9ubHkiLCJJc0ZpeGVkU2l6ZSIsIklzU3luY2hyb25pemVkIi
wiS2V5cyIsIlZhbHVlcyIsIlN5bmNSb290IiwiQ291bnQiOiJCYWxzZSIsIk
REST APIで扱える文字列の形式に変換された添付ファイル
LXhhc2h0YWJsZStWYWx1ZUNvbGxlY3Rpb24iLCJTeXN0ZW0uT2JqZWN
N0IiwiMyI=
```

DreamArts

STEP 01 連携データ取得 | STEP 02 文書情報取得 | STEP 03 新規添付ファイル追加 | STEP 04 ファイルID作成 | STEP 05 文書更新

文書情報（文書番号、既存ファイルのID）を取得



連携元データ

```
{
  "id": "0001"
  "tytle" : "〇〇申請",
  "fileId" : "xxxxxxxxxxxxxxxxxxx"
}
```

SmartDBの文書一覧のデータ

```
({
  "id": 1,
  items : {"id" : "0001"
           "tytle" : "〇〇申請",
           "file" : "xxxxxxxxxxxxxxxxxxx"
         }
},{
 【中略】
})
```

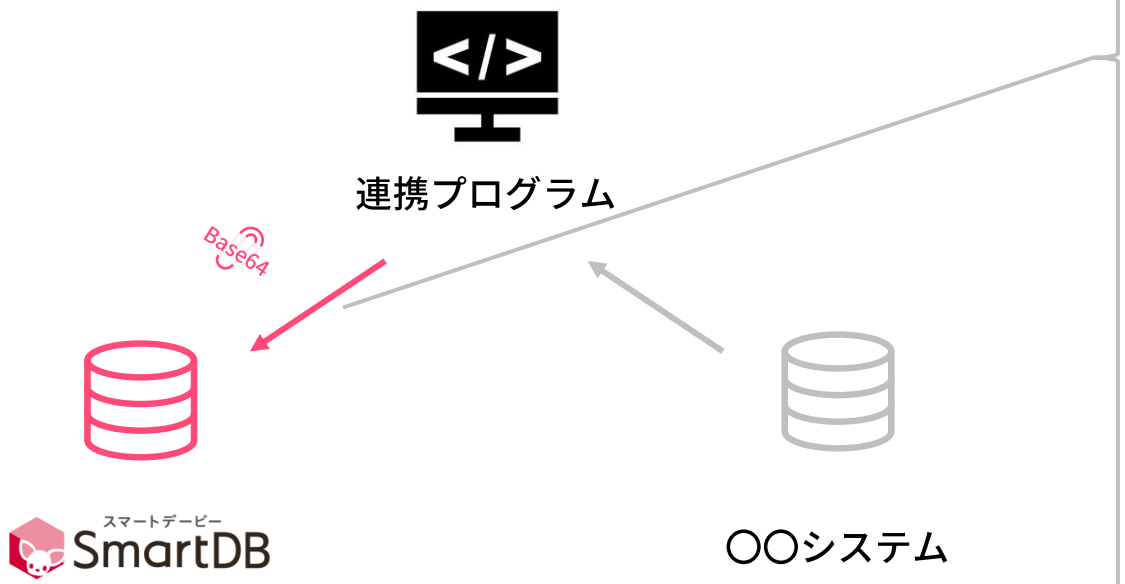
一意のデータで検索して
文書番号、ファイルIDを取得

添付ファイルのアップロード

DreamArts

STEP 01 連携データ取得 → STEP 02 文書情報取得 → **STEP 03 新規添付ファイル追加** → STEP 04 ファイルID作成 → STEP 05 文書更新

SmartDB REST API「添付ファイル追加」を実行



リクエスト

HTTPメソッド / リソースパス

POST /attaches

リクエストヘッダ

Authorization：Bearer バインダトークン
Content-Type：application/json

リクエストボディ

fileName ：ファイル名
content ：Base64でエンコードした添付ファイル

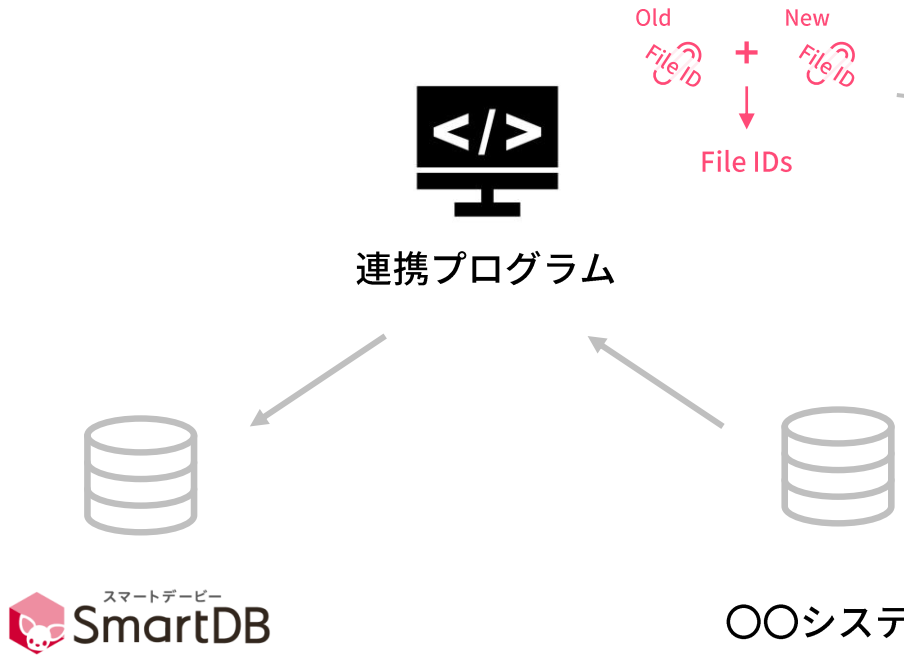
レスポンスで追加した添付ファイルのファイルIDを取得

添付ファイルのアップロード

DreamArts

STEP 01 連携データ取得 → STEP 02 文書情報取得 → STEP 03 新規添付ファイル追加 → **STEP 04 ファイルID作成** → STEP 05 文書更新

STEP02/STEP03で取得したファイルIDを1つの文字列に統合



既存ファイルのID

```
10001:1:10010:1¥n10001:1:10010:2
```

＋

新規ファイルのID

```
$$$725084267534816881.dat¥n$$$1151624727317038268O.dat
```

文書更新用のファイルID

```
10001:1:10010:1¥n10001:1:10010:2
¥n$$$725084267534816881.dat¥n$$$1151624727317038268O.dat
```

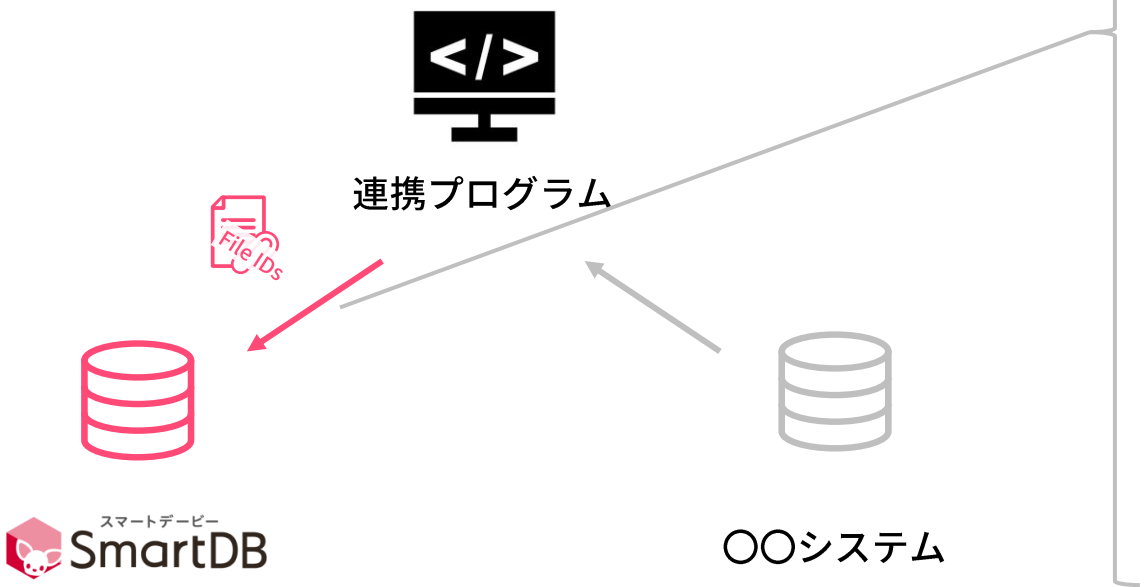
ファイルIDごとに改行コード「¥n」で繋いで統合

添付ファイルのアップロード

DreamArts

STEP 01 連携データ取得 → STEP 02 文書情報取得 → STEP 03 新規添付ファイル追加 → STEP 04 ファイルID作成 → **STEP 05 文書更新**

SmartDB REST API「文書更新」を実行



リクエスト

HTTPメソッド / リソースパス

PUT /binders/{binder}/documents/{document}

リクエストヘッダ

Authorization：Bearer バインダトークン

Content-Type：application/json

リクエストボディ

ファイル部品 部品ID ：STEP04で作成したファイルID

問題 既存文書にファイル添付する際、文書情報を取得する理由として正しいものは？

1. 既に文書に添付されているファイルのファイルIDを取得するため
2. 新たに追加するファイルのファイルIDを取得するため
3. 新規ファイルをSmartDBに追加するため
4. 特に理由はない

問題　既存文書にファイル添付する際、文書情報を取得する理由として正しいものは？

1. **既に文書に添付されているファイルのファイルIDを取得するため**
2. 新たに追加するファイルのファイルIDを取得するため
3. 新規ファイルをSmartDBに追加するため

解説

1. 既存のファイルのファイルIDを取得するため
   ※既存のファイルを考慮しなかった場合、新規ファイルのみ添付された状態となってしまうため要注意

DreamArts

問題

REST APIでファイルを既存文書に添付する方法の説明として正しいものは？

※ SmartDB REST API（v3）「 添付ファイル追加」、「文書更新」を調べながら回答してください

1. 「添付ファイル追加」では、どの文書にファイルを添付するか指定する必要がある
2. 「文書更新」では、リクエストボディで追加したいファイルのファイルIDを指定する
3. 「添付ファイル追加」では、 Base64エンコードしたファイルを直接文書に添付できる
4. 「文書更新」では、Base64エンコードしたファイルを直接文書に添付することができる

問題

REST APIでファイルを既存文書に添付する方法の説明として正しいものは？

※ SmartDB REST API（v3）「添付ファイル追加」、「文書更新」を調べながら回答してください

1. 「添付ファイル追加」では、どの文書にファイルを添付するか指定する必要がある
2. **「文書更新」では、リクエストボディで追加したいファイルのファイルIDを指定する**
3. 「添付ファイル追加」では、 Base64エンコードしたファイルを直接文書に添付できる

解説

1. 「添付ファイル追加」、「文書更新」だけでは、添付ファイルを文書に直接追加することはできない
2. 「添付ファイル追加」でBase64エンコードされたファイルをSmartDBに追加し、ファイルIDと「文書更新」を利用して文書にファイルを添付する

添付ファイルのアップロード：PowerShellで実践

添付ファイルのアップロード：PowerShellのスクリプト

```
# 0.共通情報
$sdbUri = "https://seminar.smartdb.jp/hibiki/rest/3"
$binderToken = "Lmsiv5fy_xRNLh1FCXE74fAOpw7bm57o"
$binderId = "13351"
$fileItemId = "files"
$docId = "2"
$relDirPath = "C:¥Users¥中川嵩之¥Desktop¥サンプルファイル"

# 1.既存添付ファイルの処理
$uri1 = "$sdbUri/binders/$binderId/documents/$docId"
$method1 = "GET"
$headers1 = @{
  "Authorization" = "Bearer $binderToken"
  "Content-Type" = "application/json"
}
$res1 = Invoke-WebRequest $uri1 -Method $method1 -Headers $headers1 -UseBasicParsing |
ConvertFrom-Json

$fileItem = $res1.items | Where-Object { $_.id -eq $fileItemId }
$currentFileIds = $fileItem.value | ForEach-Object { $_.id }

# 3.新規添付ファイルの処理
$filesToPost = Get-ChildItem $relDirPath -File | ForEach-Object { @{
  "fileName" = $_.Name
  "content" = [Convert]::ToBase64String([System.IO.File]::ReadAllBytes($_.FullName))
}}

$uri2 = "$sdbUri/attaches"
$method2 = "POST"
$headers2 = @{
  "Authorization" = "Bearer $binderToken"
  "Content-Type" = "application/json"
}
$body2 = @{
  "files" = $filesToPost
} | ConvertTo-Json
$body2ToBytes = [System.Text.Encoding]::UTF8.GetBytes($body2)
$res2 = Invoke-WebRequest $uri2 -Method $method2 -Headers $headers2 -Body $body2ToBytes -
UseBasicParsing | ConvertFrom-Json

$addedFileIds = $res2 | ForEach-Object { $_.id }

# 4.既存/新規ファイルのファイルID配列を結合（String形式）
$newFileIdsStr = ($currentFileIds + $addedFileIds) -join "`n"

# 5.文書データの更新：SmartDB REST API「文書更新」 ※文書番号を取得済みの前提
$uri3 = "$sdbUri/binders/$binderId/documents/$docId"
$method3 = "PUT"
$headers3 = @{
  "Authorization" = "Bearer $binderToken"
  "Content-Type" = "application/json"
}
$body3 = @{
  $fileItemId = $newFileIdsStr
} | ConvertTo-Json
Invoke-WebRequest $uri3 -Method $method3 -Headers $headers3 -Body $body3 -UseBasicParsing -
OutFile .¥response_PUTDoc.json
```

添付ファイルのアップロード：PowerShellのスクリプト

DreamArts

STEP 00　共通情報整理

共通利用する情報を変数に格納

```
# 0.共通情報
$sdbUri = "https://【sdbDmain】.smartdb.jp/hibiki/rest/3"
$binderToken = "【binderToken】"
$binderId = "【binderId】"
$fileItemId = "【itemId】"
$docId = "【docId】"
$relDirPath = "【folderPath】"
```

STEP 01　文書情報取得

既存文書の情報（文書番号、ファイルID）を取得

ファイルID配列の作成（既存）

```
# 1.既存添付ファイルの処理
$uri1 = "$sdbUri/binders/$binderId/documents/$docId"
$method1 = "GET"
$headers1 = @{
  "Authorization" = "Bearer $binderToken"
  "Content-Type" = "application/json"
}
$res1 = Invoke-WebRequest $uri1 -Method $method1 -Headers $headers1 -UseBasicParsing | ConvertFrom-Json

$fileItem = $res1.items | Where-Object { $_.id -eq $fileItemId }
$currentFileIds = $fileItem.value | ForEach-Object { $_.id }
```

STEP 02　新規添付ファイル追加

添付するファイルをSmartDBに追加

ファイルID配列の作成（新規）

```
# 3.新規添付ファイルの処理
$filesToPost = Get-ChildItem $relDirPath -File | ForEach-Object { @{
  "fileName" = $_.Name
  "content" = [Convert]::ToBase64String([System.IO.File]::ReadAllBytes($_.FullName))
}}

$uri2 = "$sdbUri/attaches"
$method2 = "POST"
$headers2 = @{
  "Authorization" = "Bearer $binderToken"
  "Content-Type" = "application/json"
}
$body2 = @{
  "files" = $filesToPost
} | ConvertTo-Json
$body2ToBytes = [System.Text.Encoding]::UTF8.GetBytes($body2)
$res2 = Invoke-WebRequest $uri2 -Method $method2 -Headers $headers2 -Body $body2ToBytes -UseBasicParsing | ConvertFrom-Json

$addedFileIds = $res2 | ForEach-Object { $_.id }
```

STEP 03　ファイルID作成

既存/新規のファイルIDを1つの文字列に統合

```
# 4.既存/新規ファイルIDを合体（1つの文字列に）
$newFileIdsStr = ($currentFileIds + $addedFileIds) -join "`n"
```

STEP 04　文書更新

ファイルIDでファイルフォルダ部品の値を更新

```
# 5.文書データの更新：SmartDB REST API「文書更新」 ※文書番号を取得済みの前提
$uri3 = "$sdbUri/binders/$binderId/documents/$docId"
$method3 = "PUT"
$headers3 = @{
  "Authorization" = "Bearer $binderToken"
  "Content-Type" = "application/json"
}
$body3 = @{
  $fileItemId = $newFileIdsStr
} | ConvertTo-Json
Invoke-WebRequest $uri3 -Method $method3 -Headers $headers3 -Body $body3 -UseBasicParsing -OutFile .¥response_PUTDoc.json
```

DreamArts

```
# 0.共通情報
$sdbUri = "https://seminar.smartdb.jp/hibiki/rest/3"
$binderToken = "Lmsiv5fy_xRNLh1FCXE74fAOpw7bm57o"
$binderId = "13351"
$fileItemId = "files"
$docId = "2"
$relDirPath = "C:¥Users¥中川嵩之¥Desktop¥サンプルファイル"

# 1.既存添付ファイルの処理
$uri1 = "$sdbUri/binders/$binderId/documents/$docId"
$method1 = "GET"
$headers1 = @{
  "Authorization" = "Bearer $binderToken"
  "Content-Type" = "application/json"
}
$res1 = Invoke-Web
ConvertFrom-Json

$fileItem = $res1.items | Where-Object { $_.id -eq $fileItemId }
$currentFileIds = $fileItem.value | ForEach-Object { $_.id }

# 3.新規添付ファイルの処理
$filesToPost = Get-ChildItem $relDirPath -File | ForEach-Object { @{
  "fileName" = $_.Name
  "content" = [Convert]::ToBase64String([System.IO.File]::ReadAllBytes($_.FullName))
}}

$uri2 = "$sdbUri/attaches"
$method2 = "POST"
$headers2 = @{
  "Authorization" = "Bearer $binderToken"
  "Content-Type" = "application/json"
}
$body2 = @{
  "files" = $filesToPost
} | ConvertTo-Json
$body2ToBytes = [System.Text.Encoding]::UTF8.GetBytes($body2)
$res2 = Invoke-WebRequest $uri2 -Method $method2 -Headers $headers2 -Body $body2ToBytes -
UseBasicParsing | ConvertFrom-Json

$addedFileIds = $res2 | ForEach-Object { $_.id }

# 4.既存/新規ファイルのファイルID配列を結合（String形式）
$newFileIdsStr = ($currentFileIds + $addedFileIds) -join "`n"

$headers3 = @{
  "Authorization" = "Bearer $binderToken"
  "Content-Type" = "application/json"
}
$body3 = @{
  $fileItemId = $newFileIdsStr
} | ConvertTo-Json
Invoke-WebRequest $uri3 -Method $method3 -Headers $headers3 -Body $body3 -UseBasicParsing -
OutFile .¥response_PUTDoc.json
```

詳細なスクリプトの解説はサポートサイトに掲載予定

DreamArts

1. 前回の復習
2. 応用的なSmartDB REST API
3. まとめ
4. 次回予告

1. SmartDB REST APIの組み合わせには**要件ごとにパターン**がある
2. マスタバインダの同期では、「**CSV入力（更新）**」を利用する
3. マスタバインダの同期方法には**UPSERT**と**REPLACE**がある
4. 添付ファイルのULでは、「**添付ファイル追加**」、「**文書更新**」を利用する
5. 添付ファイルのULでは、**既存文書の添付ファイルを考慮**する必要がある

DreamArts

1. 前回の復習
2. 応用的なSmartDB REST API
3. まとめ
4. 次回予告

次回予告

DreamArts

No.	実施概要	詳細内容	推奨者
1	REST API 入門編	REST API概説 実践：PowerShellでREST APIを実行してみよう	SmartDB利用経験のある方
2	SmartDB REST API アプリ操作基本編	SmartDB REST APIの基本的な使い方 よく使う基本的なAPIの解説と実践	1の参加者、もしくは1の内容を理解している方
本日	SmartDB REST API アプリ操作応用編	SmartDB REST APIの応用的な使い方 複数のREST APIを用いた業務適用の解説と実践	2の参加者、もしくは2の内容を理解している方
次回	SmartDB REST API アカウンタマスタ連携編	アカウントAPIの使い方 アカウントAPIを用いた業務適用の解説と実践	1の参加者、もしくは1の内容を理解している方
5	SmartDB REST API 業務・性能・セキュリティ編	業務影響を軽減するための設計 SDBの性能を考慮した設計・運用	3の参加者、もしくは3の内容を理解している方 または、 4の参加者、もしくは4の内容を理解している方
6	SmartDB REST API システム連携設計編	SmartDBを含む複数システム連携の全体設計 SDB起点の実行方法：Webhook、定期バッチ処理	5の参加者、もしくは5の内容を理解している方

※上記の内容は変更される場合がございます

お知らせ

DreamArts

資料や動画は、後日コミュニティサイト「スマラジ！ルーム」へ掲載します。

本イベントに関するご質問も受付けますのでお気軽に投稿ください。

https://cs.support-smartdb.com/hc/ja/community/topics/5423952120601

※閲覧にはサポートサイトへのログインが必要です。

DreamArts

コミュニティサイトでは、DAや他ユーザーさんへの質問が可能です。
気になったことがあれば、お気軽に投稿ください。

スマートデービー SmartDB サポートサイト

サイト内検索

SmartDBサポートサイト ＞ コミュニティ ＞ 雑談・つぶやき

非推奨のログイン方法だけど・・・SmartDB REST APIをExcel VBAで試した話

あおさん（趣味C#プログラマー） Great supporter First post

2023年09

前回に続いて、API関連の話を書いてみます。
但し、今回のログイン方法は**公式に非推奨**であり、
セキュリティ的によろしくなく、
またいつ使えなくなってもおかしくないのでご注意ください。
また、複雑になるのでVBAのコードは書きません。

まず、最新のAPI（V3）を確認してみます。

SmartDB REST API (support-dreamarts.com)

セッションについての説明の3つ目を見ます

- `post /session` を実行して、セッションを作成する。※非推奨

  ID/PASSWORDを指定してsession APIを実行してセッションを作成します。
  ここで作成したセッションを利用して、その他のREST APIを実行します。バッチ処理などを特定ユーザで実行したい場合など。
  プログラム中にユーザのID/パスワードを記述する必要があるため、セキュリティの観点からも非推奨とします。

正直この説明が全てです。
バインダAPIオプション契約前のテストぐらいには使えるかもしれませんが、
セキュリティ的に本運用には使えません。

まず、この方法がまだ使えるのか確認します。
お手軽にWindowsのコマンドプロンプトからcurlでやってみます。
ログインが、

SmartDBサポートサイト ＞ コミュニティ ＞ 一般Q&A

グラフダイアログのカラー設定

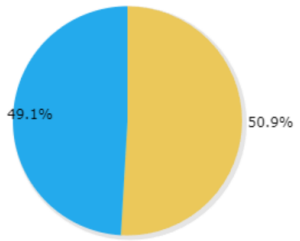
のほほん Great supporter Nice multiple posts

202

皆さん、グラフダイアログって利用されていますか？
簡易なグラフなら便利かな？と思って設定していたのですが
値の大きい順に色設定されてしまう事に気が付きました。
これって、あたり前の事なのでしょうか(゜Д゜)

車の運転前と運転後にそれぞれ登録するバインダがあります。
運転前の登録数と運転後の登録数は原則イコールになる想定なのですが
明らかに登録割合のおかしい人には注意をしようと思ってグラフを表示しました。

●Aさん



運転前後	部品件数	比率
運転後	55	50.9%
運転前	53	49.1%
合計	108	100.0%

運転前と運転後がほぼ同じなので、よし

フォローする

SmartDBサポートサイト ＞ コミュニティ ＞ What's New！

スマコミュ！マガジン_vol.31～10/18デジタルの民主化DAYを開催しました！～

フォローする

廣瀬 璃奈 DreamArts

2023年11月09日 14:14

0

■◆■――――――――――――――― vol.31―2023.11.09―
スマコミュ！マガジン
―――――――――――――――――――――――――■◆■

こんにちは！ドリーム・アーツの廣瀬です。

涼しくなったかと思えば急に気温が上がったりと、なかなか不安定な時期が続いていますね。
私は週末に少し遠出をして紅葉を見に行きました🍁木々の葉っぱの色が変わっていく瞬間は、季節の変化を感じます。
皆さんが季節の変わり目を感じる瞬間はいつでしょうか？

今月もSmartDBお役立ち情報を配信していきます。

――本日のお役立ち情報――――――――――――――――――――――――

1. 10/18デジタルの民主化DAYを開催しました！
2. 10月のコミュニティサイト人気投稿ランキング
3. SmartDBのレビューを投稿してAmazonギフト券をゲット！

―――――――――――――――――――――――――――――――――――――

DreamArts

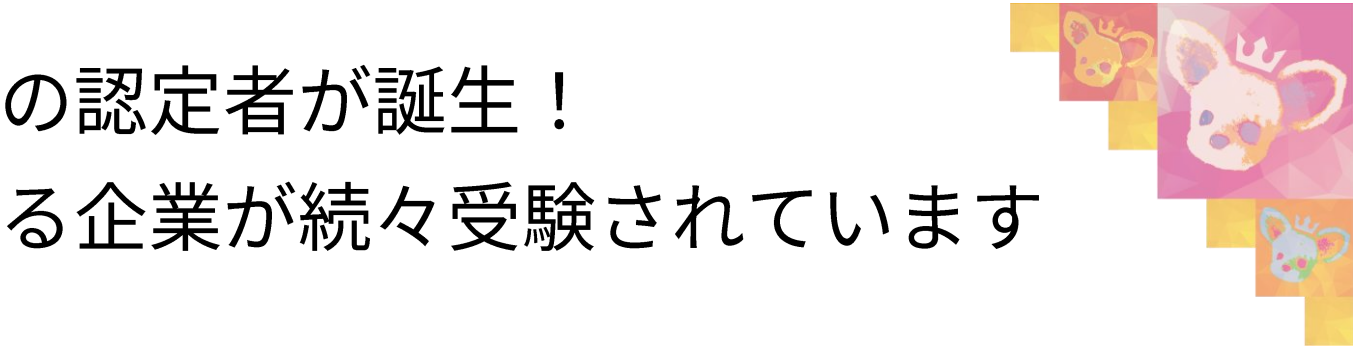
SmartDB認定制度

SmartDB Certified Specialist

SmartDBによる
業務デジタル化の習熟度・スキルを
個人へ認定するプログラム

詳細はこちら ▶ https://hibiki.dreamarts.co.jp/smartdb/scs/

DreamArts



すでに、多くの認定者が誕生！
デジタルの民主化を推進する企業が続々受験されています

JAPAN AIRLINES
Daiwa House 大和ハウスグループ
YAMADA HLDGS.
DLS Dai-ichi Life Group
三菱オートリース
SEKISUIKASEI
RITSUMEIKAN
sala サーラグループ
CARTA
HITOWA GROUP
OYO 応用地質株式会社
LIXIL トータルサービス
CAPCOM
アイダ設計
Tomorrow, Together KDDI

※弊社サイトより抜粋

【3月31日まで】コミュニティ参加者限定

ITreviewへのアンケート掲載でもれなくプレゼント！

Amazonギフト券 3,000円分

スマートデービー SmartDB × ITreview

キャンペーンコードを必ずご入力ください

AUQBMHOY

※「キャンペーン要項」を必ずご確認ください

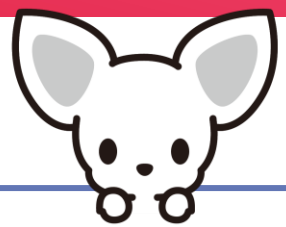
無くなり次第終了

Step.1 ITreviewに会員登録をしよう

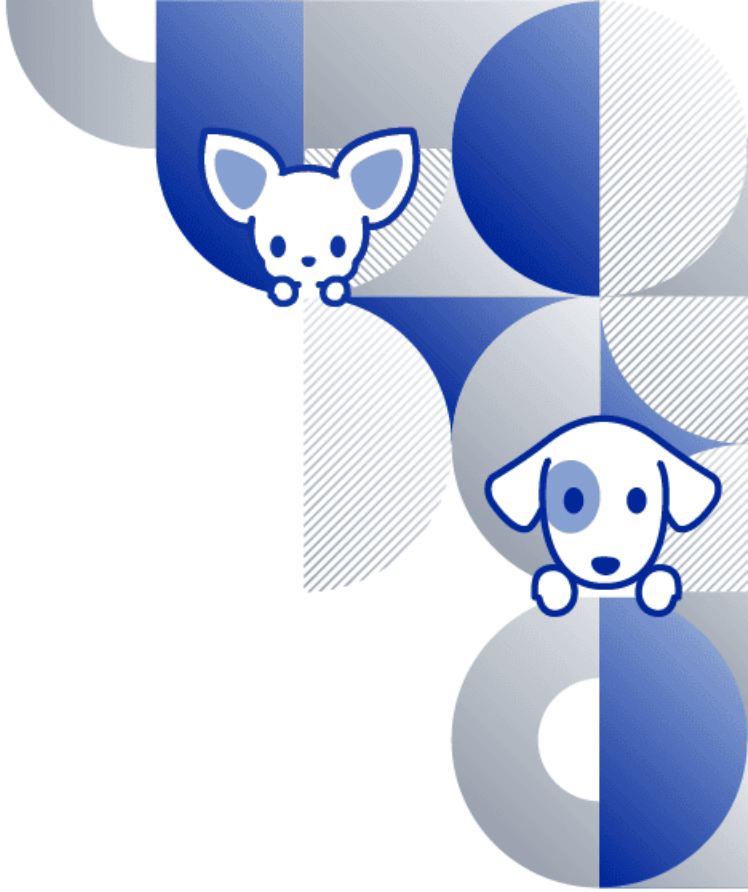
Step.2 レビュー・口コミを書こう

Step.3 掲載されると…

Step.4 Amazonギフト券GET!!



キャンペーン専用ページURL



DreamArts

https://www.dreamarts.co.jp/