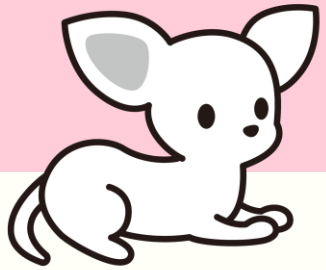
スマラジ！ SmartDB Radio

活用レベルアップ！

SmartDB深掘りセミナー



基礎を学ぼう！REST API入門編

～他システム連携方法を全6回で習得！①～

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 CSX2G
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：　廣瀬 璃奈（ひろせ りな）
所属：　協創パートナー推進本部 CSX2G
出身：　東京都
経歴：　元投資系IT会社で営業支援
　　現在はコミュニティ運営担当
趣味：　サウナ、ドライブ、美食巡り

DreamArts

1. REST APIとは？
2. REST APIの利用方法と実践
3. まとめ
4. 次回予告

DreamArts

1. REST APIとは？
2. REST APIの利用方法と実践
3. まとめ
4. 次回予告

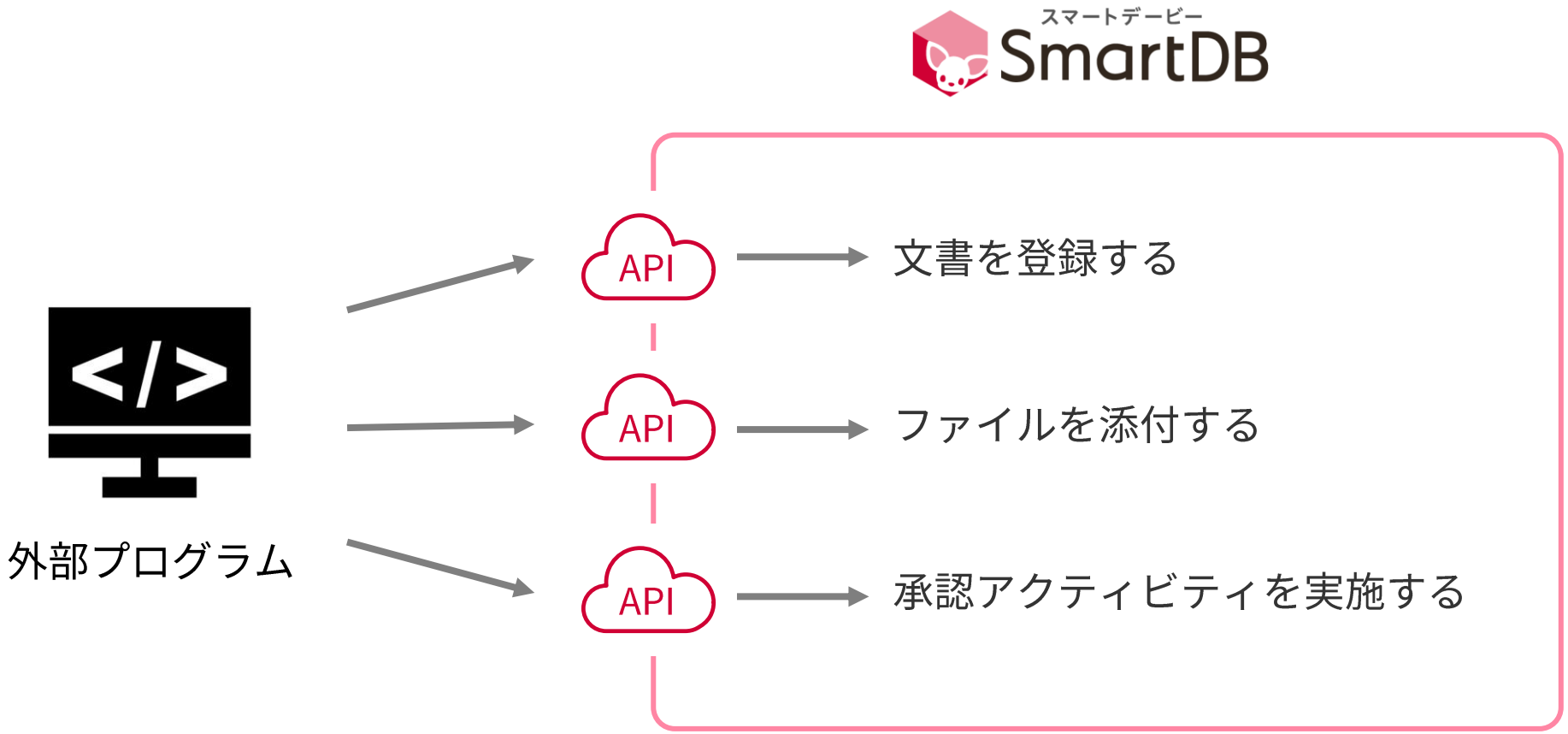
REST API

Application Programming Interface

『外部（のプログラム）から機能・情報を利用するための窓口』

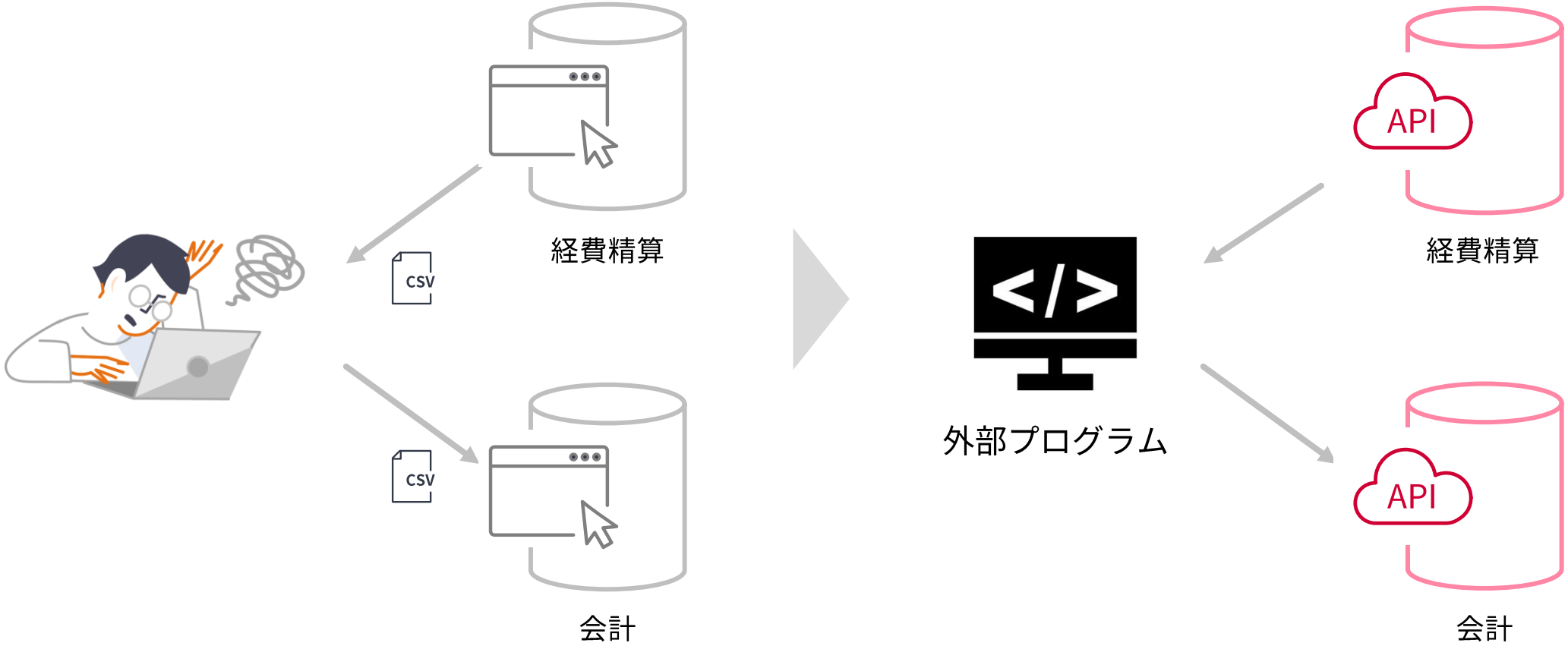
DreamArts

SmartDBの持つ機能を**外部プログラム**で利用できる



DreamArts

外部プログラムを介して、**システム間の自動連携**を実現できる



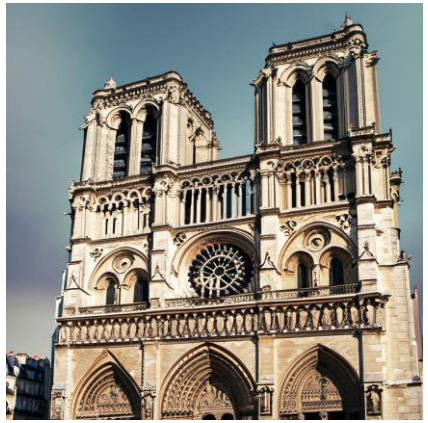
REST API

REpresentational State Transfer

『“RESTの原則”という設計モデルに基づいて作られているAPI』

※SmartDBを含むWebサービスで広く使われている設計モデル

DreamArts



ゴシック様式

- 高い天井と尖塔
- 垂直性と光を重視
- 装飾的な彫刻

ルネサンス様式

- 大きなドーム構造
- 幾何学的
- 平穏な装飾

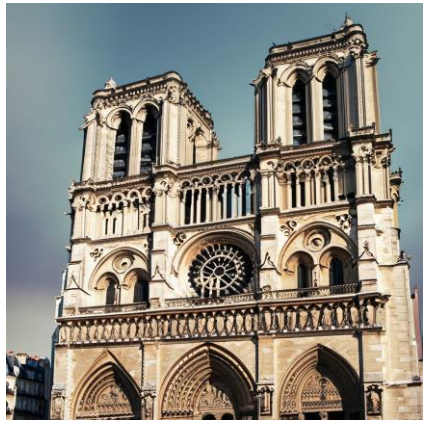
REST API

- シンプルで軽量
- 柔軟性が高い
- 比較的簡単

SOAP API

- 機能が豊富
- 厳格で堅固
- 複雑性が高い

DreamArts



ゴシック様式

- 高い天井と尖塔
- 垂直性と光を重視
- 装飾的な彫刻

ルネサンス様式

- 大きなドーム構造
- 幾何学的
- 平穏な装飾

REST API

- シンプルで軽量
- 柔軟性が高い
- 比較的簡単

SOAP API

- 機能が豊富
- 厳格で堅固
- 複雑性が高い

DreamArts

ゴシック様式

- 高い天井と尖塔
- 垂直性と光を重視
- 装飾的な彫刻

ルネサンス様式

- 大きなドーム構造
- 幾何学的
- 平穏な装飾

REST API

- シンプルで軽量
- 柔軟性が高い
- 比較的簡単

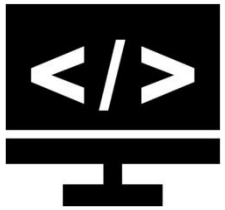
SOAP API

- 機能が豊富
- 厳格で堅固
- 複雑性が高い

DreamArts

外部プログラムとWebサービス間で2つの通信が行われる

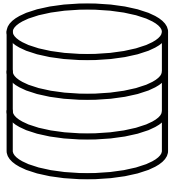
顧客マスタにある顧客IDがA00001の
顧客名を教えてください！



外部プログラム

リクエスト（要求）

レスポンス（応答）



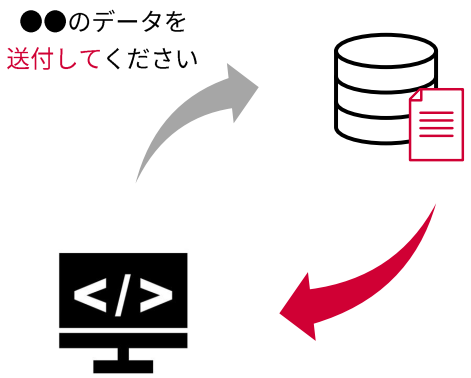
Webサービス

了解です！
顧客名は株式会社ドリーム・アーツです。

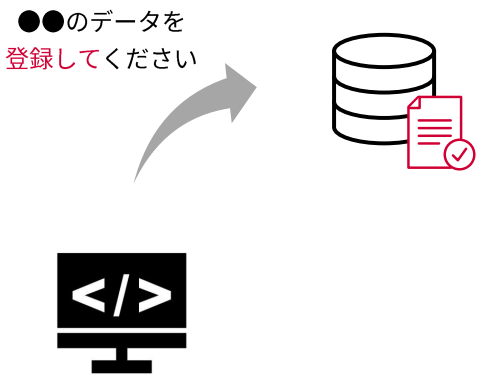
DreamArts

Webサービスに対して主に4つの処理を要求できる

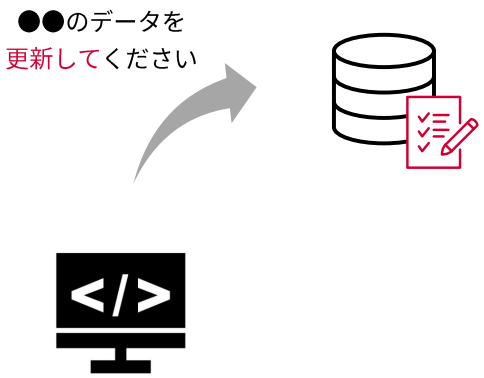
取得



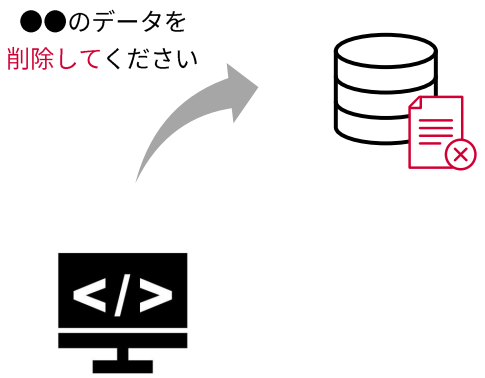
登録



更新



削除



DreamArts

1 APIとは、外部から**機能・情報を利用するための窓口**である

2 REST APIには、**リクエスト**と**レスポンス**という2つの通信がある

3 REST APIでは、データの**取得**、**登録**、**更新**、**削除**ができる

DreamArts

1. REST APIとは？
2. REST APIの利用方法と実践
   - リクエストを作る
   - レスポンスを読む
3. まとめ
4. 次回予告

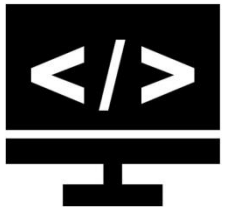
DreamArts

1. REST APIとは？
2. REST APIの利用方法と実践
   - リクエストを作る
   - レスポンスを読む
3. まとめ
4. 次回予告

DreamArts

Webサービスに対してリクエストを送る必要がある

顧客マスタにある顧客IDがA00001の
顧客名を教えてください！



外部プログラム

リクエスト（要求）

レスポンス（応答）

Webサービス

了解です！
顧客名は株式会社ドリーム・アーツです。

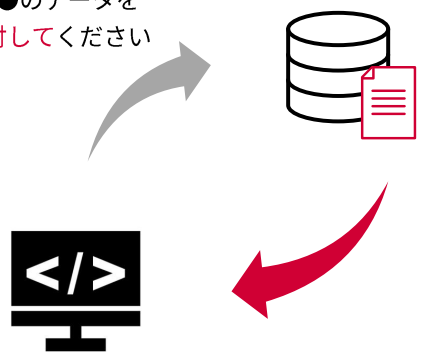
取得・削除のリクエスト

取得・削除のリクエスト

DreamArts

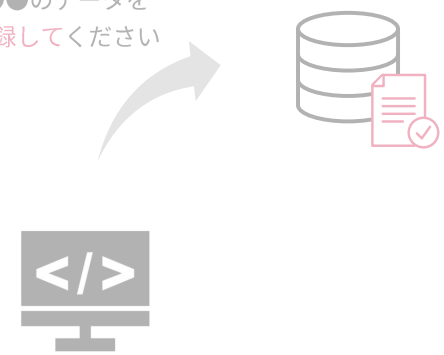
取得

●●のデータを
送付してください



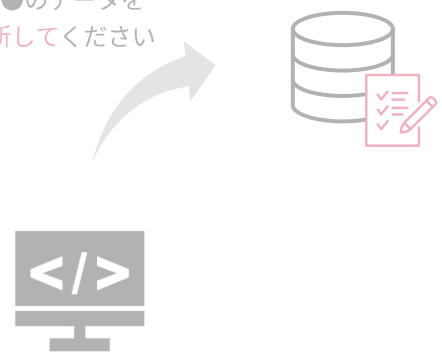
登録

●●のデータを
登録してください



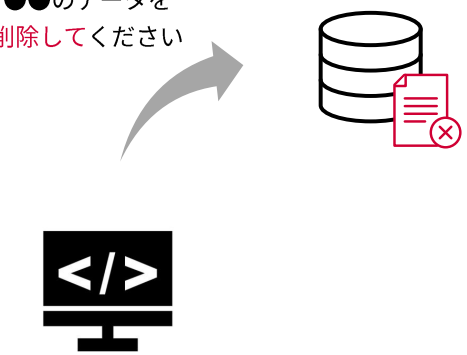
更新

●●のデータを
更新してください



削除

●●のデータを
削除してください

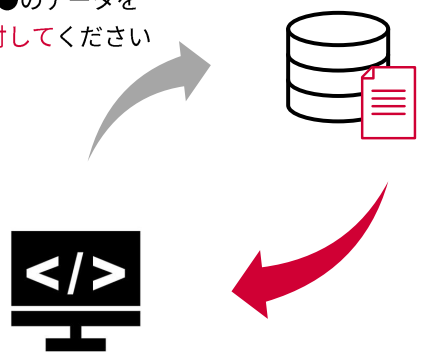


取得・削除のリクエスト

DreamArts

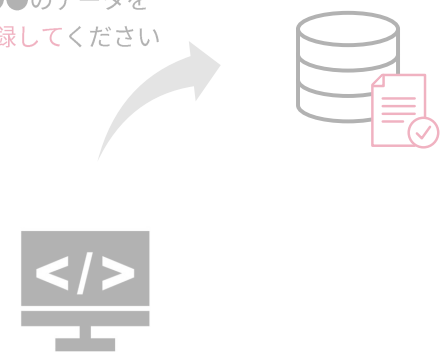
取得

●●のデータを
送付してください



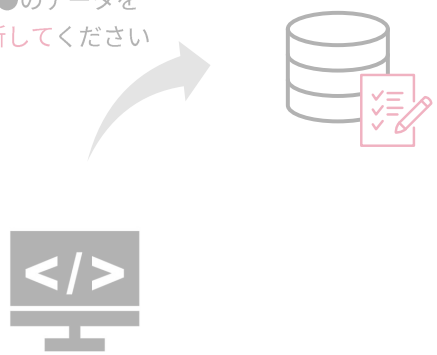
登録

●●のデータを
登録してください



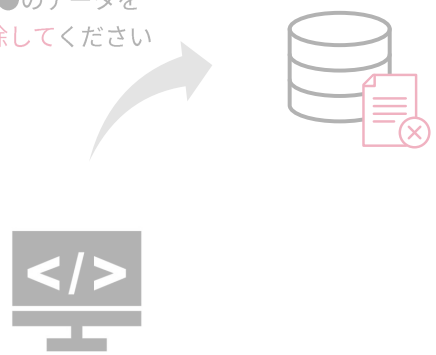
更新

●●のデータを
更新してください



削除

●●のデータを
削除してください



前提

- 営業部に所属するあなたは、新たにX社を担当することとなる
- 次回面談に向けて、あなたはX社の情報を必要としている
- 同部に所属する同僚のBさんは顧客資料を全て管理している

あなた

同僚 Bさん

あなたは、Bさんにどのような依頼をする？

？

あなた

同僚 Bさん

あなたは、Bさんにどのような依頼をする？

- 顧客資料のX社に関する情報をください
- X社の顧客資料を見せてください

あなた

同僚 Bさん

あなたは、Bさんにどのような依頼をする？

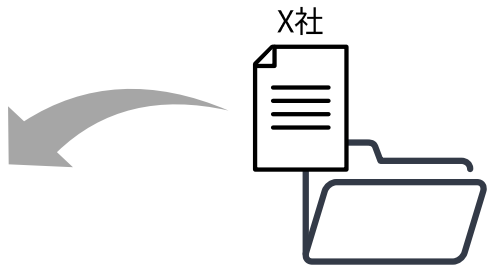
1. 顧客資料のX社に関して
2. 情報をください

あなた

同僚 Bさん

DreamArts

1. ○○というリソースに対して
   顧客資料のうちのX社の資料
2. データを××してください
   取得

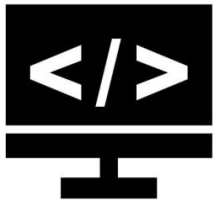


あなた

リクエスト（要求）

同僚 Bさん

1. ○○というリソースに対して
   対象のリソースを**URI**で指定
2. データを××してください
   実施する処理を**HTTPメソッド**で指定



外部プログラム

リクエスト（要求）

Webサービス

DreamArts

URIを用いて対象リソースを指定

https://HostName.jp/Resouce/Path/PathParam?Query1=True& Query2=False

① ② ③ ④

①プロトコル 必須

通信方式を指定

ほとんどの場合、「https」を利用する。

②ホスト名 必須

Webサービス・環境を識別するために指定

③リソースパス 必須

利用するREST APIを識別するために指定

URLのように階層構造で表現される。

④クエリパラーメータ

データのソート順など、必要なオプションを指定

「？」で始まり、項目と値を「=」で結ぶ。

複数の項目がある場合は、「&」で繋げる。

DreamArts

URIを用いて対象リソースを指定

https://HostName.jp/Resouce/Path/PathParam?Query1=True& Query2=False

① ② ③ ④

①プロトコル 必須

通信方式を指定

ほとんどの場合、「https」を利用する。

②ホスト名 必須

Webサービス・環境を識別するために指定

③リソースパス 必須

利用するREST APIを識別するために指定

URLのように階層構造で表現される。

④クエリパラーメータ

データのソート順など、必要なオプションを指定

「？」で始まり、項目と値を「=」で結ぶ。

複数の項目がある場合は、「&」で繋げる。

DreamArts

URIを用いて対象リソースを指定

https://HostName.jp/Resouce/Path/PathParam?Query1=True& Query2=False

① ② ③ ④

①プロトコル 必須

通信方式を指定

ほとんどの場合、「https」を利用する。

②ホスト名 必須

Webサービス・環境を識別するために指定

③リソースパス 必須

利用するREST APIを識別するために指定

URLのように階層構造で表現される。

④クエリパラーメータ

データのソート順など、必要なオプションを指定

「？」で始まり、項目と値を「=」で結ぶ。

複数の項目がある場合は、「&」で繋げる。

DreamArts

URIを用いて対象リソースを指定

https://HostName.jp/Resouce/Path/PathParam?Query1=True& Query2=False

① ② ③ ④

①プロトコル 必須

通信方式を指定

ほとんどの場合、「https」を利用する。

②ホスト名 必須

Webサービス・環境を識別するために指定

③リソースパス 必須

利用するREST APIを識別するために指定

URLのように階層構造で表現される。

④クエリパラーメータ

データのソート順など、必要なオプションを指定

「？」で始まり、項目と値を「=」で結ぶ。

複数の項目がある場合は、「&」で繋げる。

DreamArts

URIを用いて対象リソースを指定

https://HostName.jp/Resouce/Path/PathParam?Query1=True& Query2=False

① ② ③ ④

①プロトコル 必須

通信方式を指定

ほとんどの場合、「https」を利用する。

②ホスト名 必須

Webサービス・環境を識別するために指定

③リソースパス 必須

利用するREST APIを識別するために指定

URLのように階層構造で表現される。

④クエリパラーメータ

データのソート順など、必要なオプションを指定

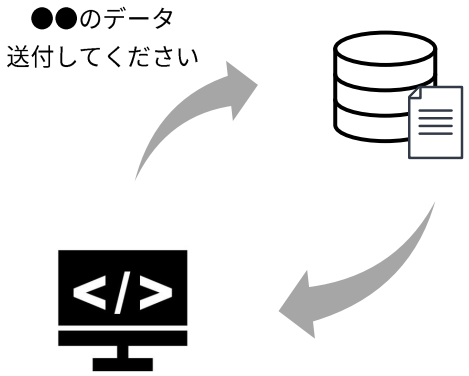
「？」で始まり、項目と値を「=」で結ぶ。

複数の項目がある場合は、「&」で繋げる。

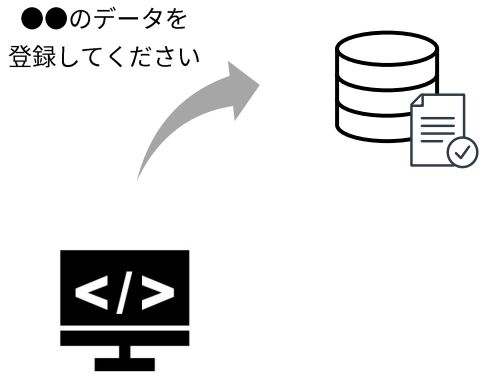
DreamArts

HTTPメソッドを用いて処理を指定

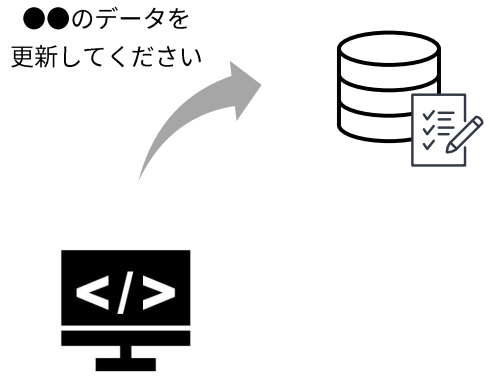
取得



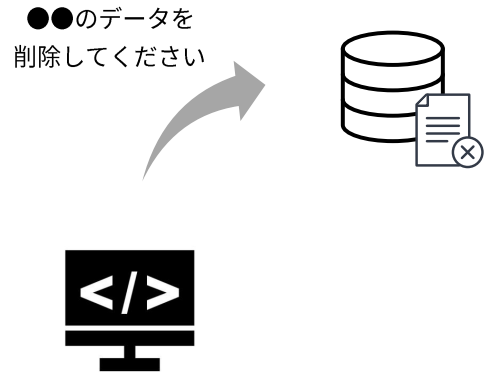
登録



更新



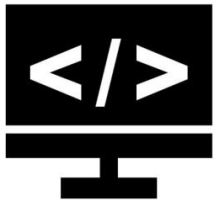
削除



DreamArts

HTTPメソッドを用いて処理を指定

1. 顧客情報バインダのX社の文書というリソースに対して
   - https://xxx.smartdb.jp/10001/documents/10
     （10001：バインダID、10：文書ID）
2. データを取得してください
   - GET



外部プログラム

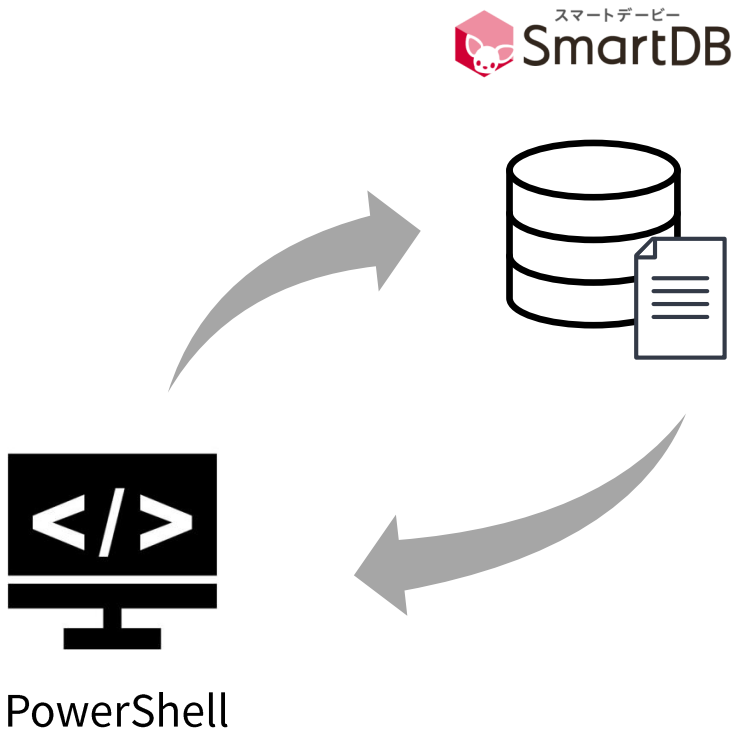
リクエスト（要求）

SmartDB

実践：PowerShellでREST APIを実行しよう

顧客マスタから文書情報を取得

1. 顧客マスタから文書情報を取得しよう



2. 顧客マスタに文書を登録しよう

取得する元データの確認

① 下記に作成されている「顧客マスタ」バインダを開く
トップ >> スマラジ！REST API >> 1.入門編

② 文書番号1の文書を開く

DreamArts

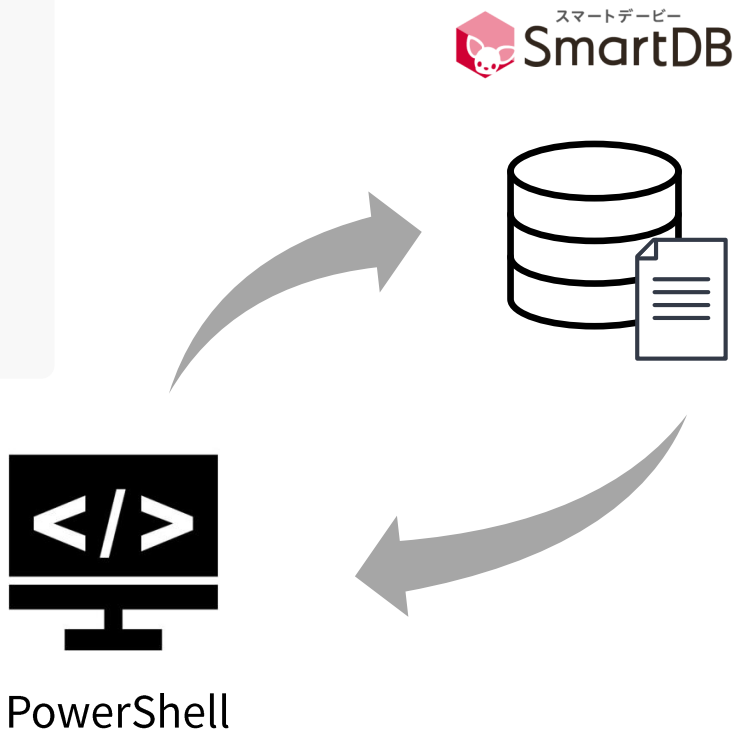
PowerShellでREST APIを実行する方法

1. URI
2. HTTPメソッド
3. リクエストボディ
4. リクエストヘッダ

の指定

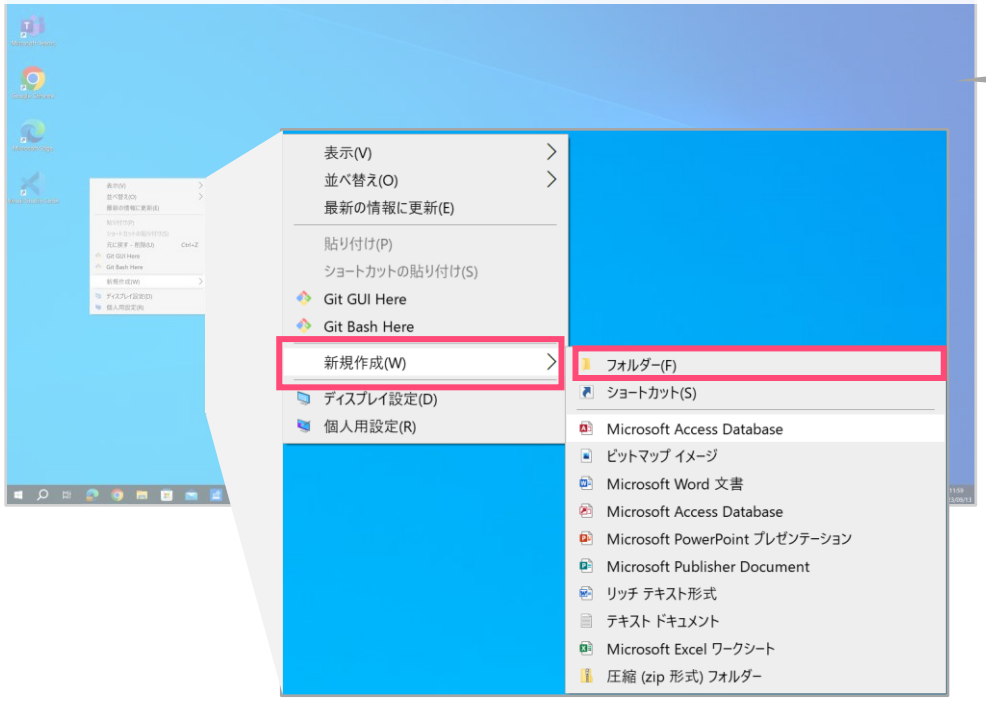
＋

Invoke-WebRequest：REST APIを実行するコマンド

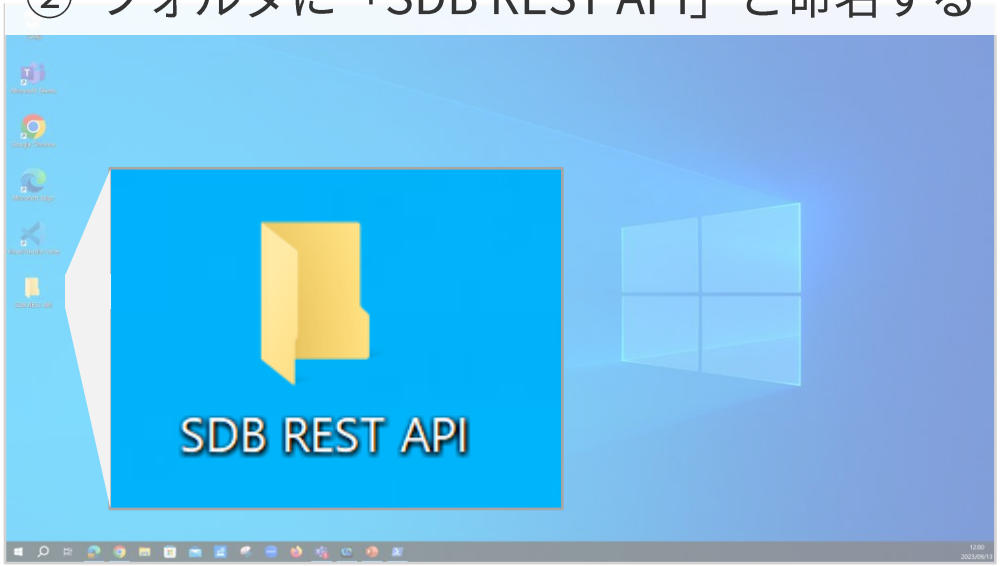


ハンズオン用のフォルダを作成する

① デスクトップ上にフォルダを作成する

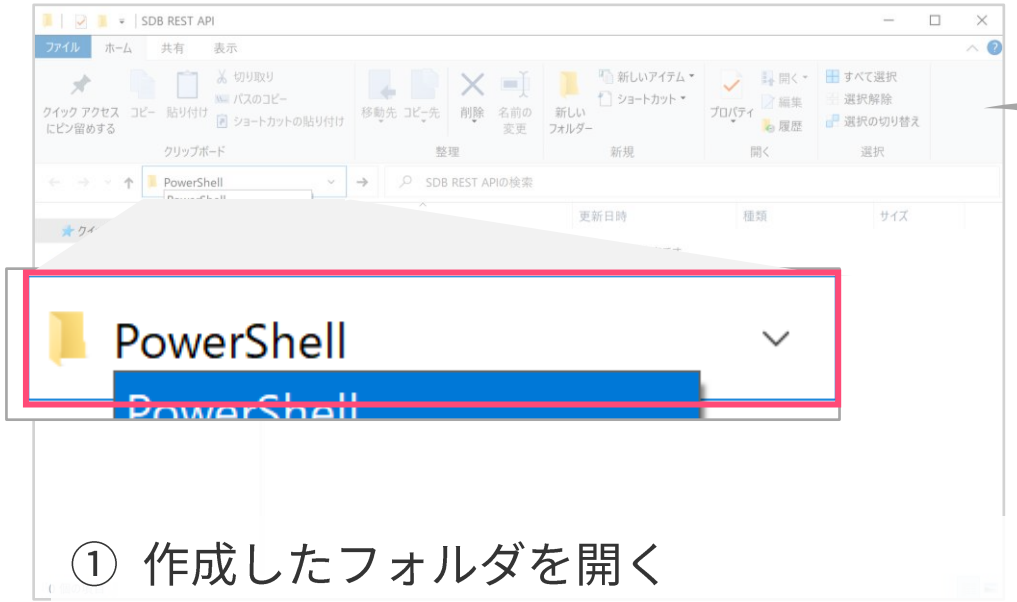


② フォルダに「SDB REST API」と命名する

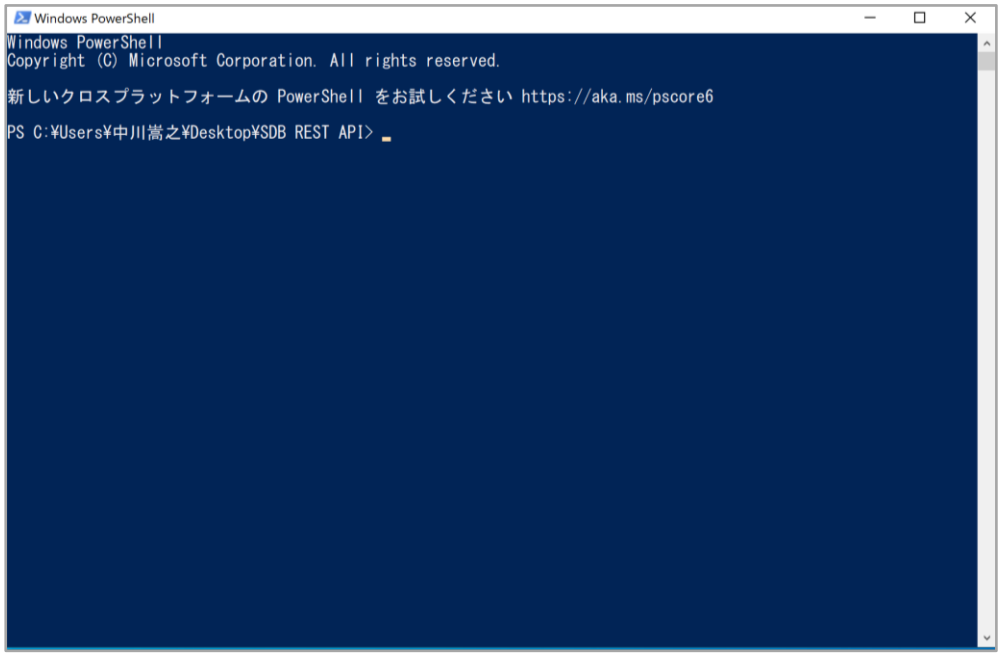


DreamArts

作成したフォルダからPowerShellを起動する



① 作成したフォルダを開く
② アドレスバーに「PowerShell」と入力する
③ Enterキーを押す

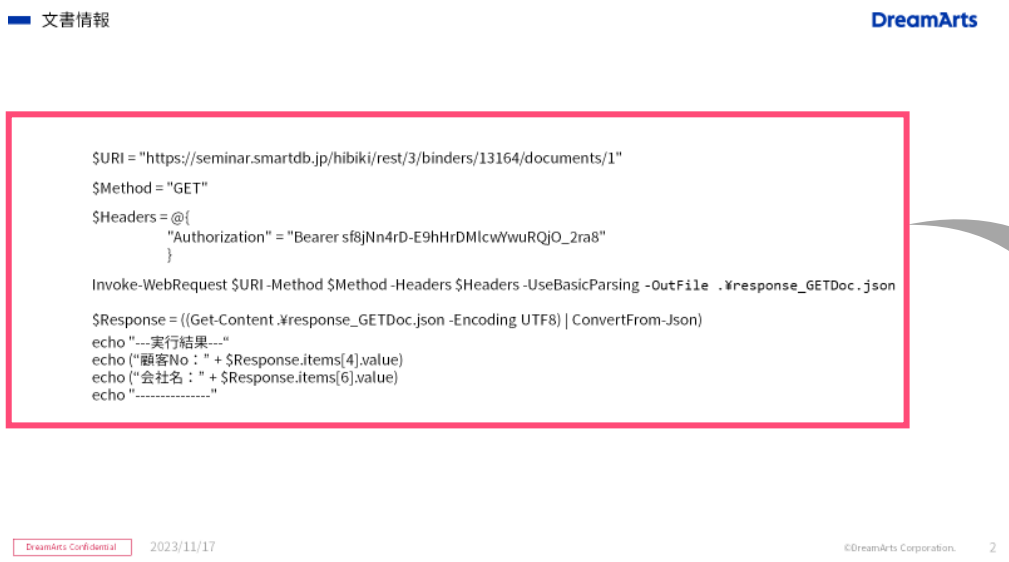


※ログファイルの出力先を今回作成したフォルダにするため

DreamArts

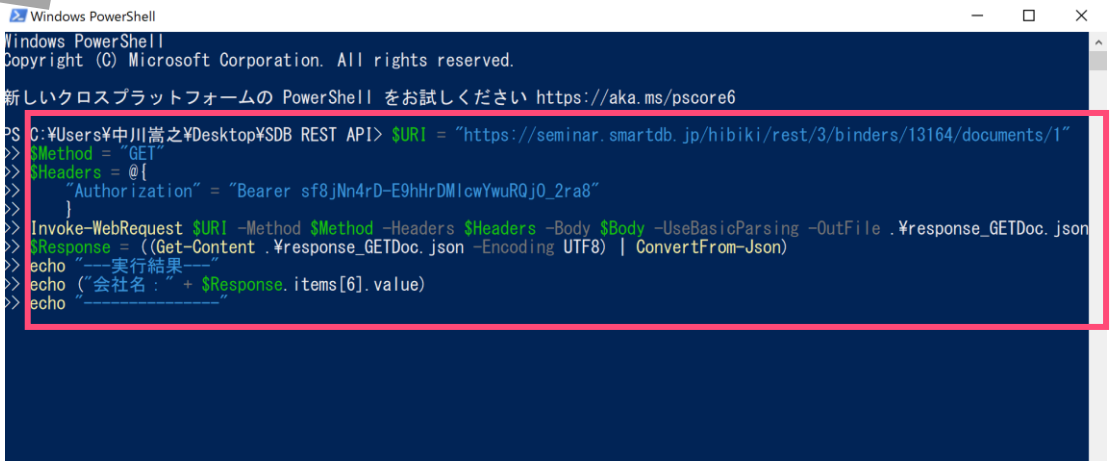
事前配布のコードをPowerShellにコピペする

① 事前に配布されたPDFの「文書情報取得」のコードをコピー



② PowerShellに貼り付け

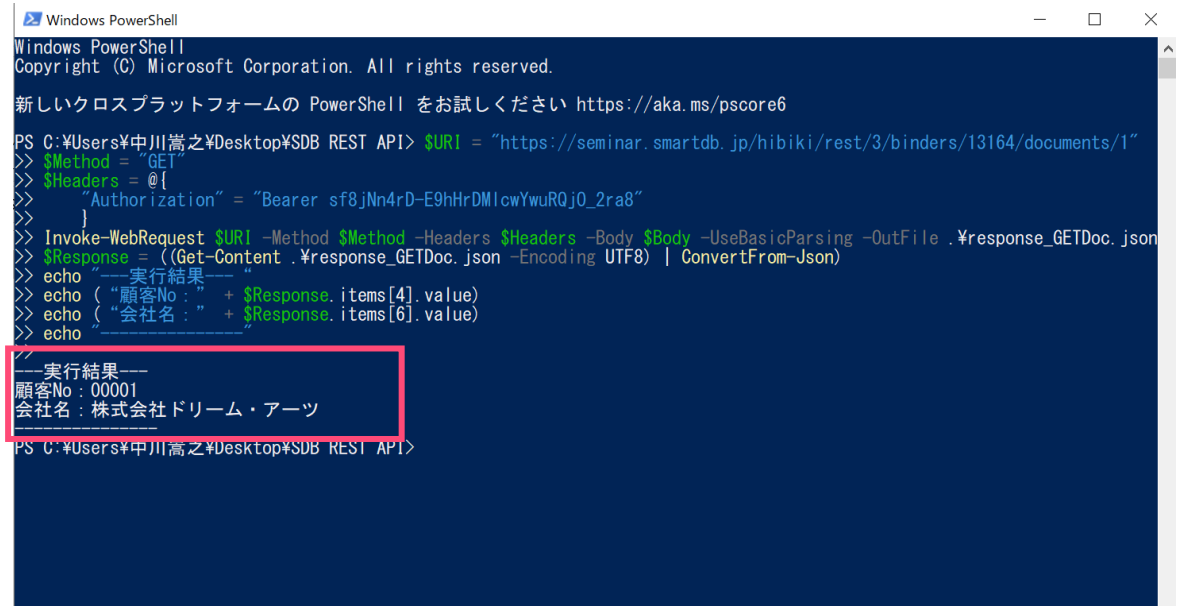
③ Enterキーで実行



DreamArts

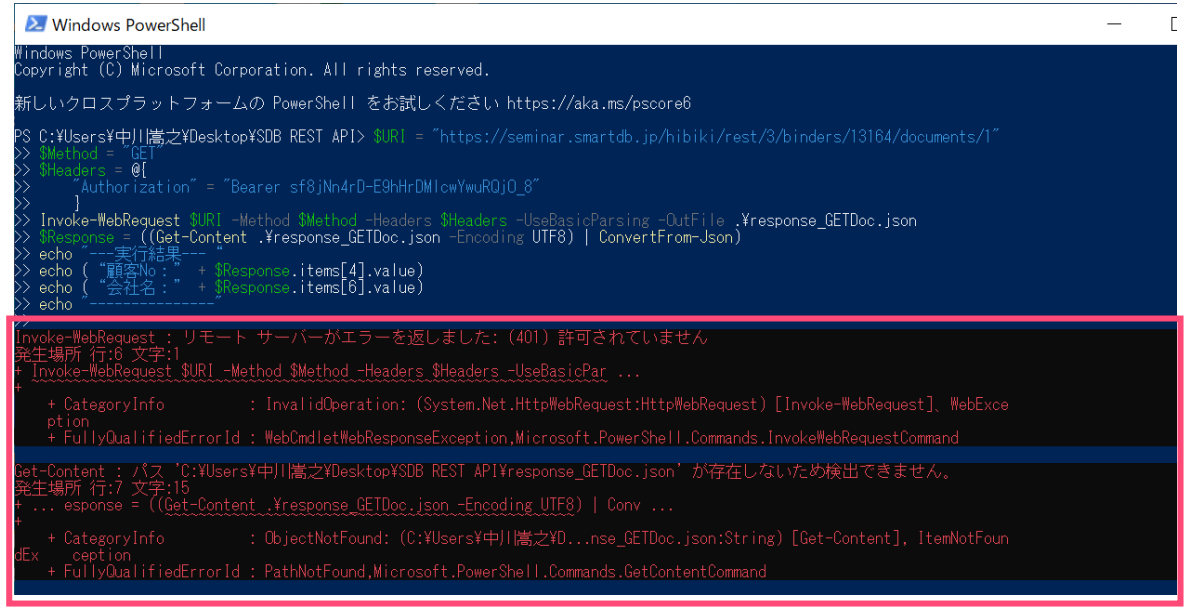
実行結果を確認する

成功



実際のデータが出力

失敗



赤字でエラーが出力

DreamArts

実行結果を確認する

DreamArts

```
①
$URI = "https://seminar.smartdb.jp/hibiki/rest/3/binders/13164/documents/1"
②
$Method = "GET"
③
$Headers = @{
            "Authorization" = "Bearer sf8jNn4rD-E9hHrDMlcwYwuRQjO_2ra8"
            }
(
(
```

解説

① プロトコル、ホスト名、リソースパスからURIを作成

プロトコル：https

ホスト名： seminar.smartdb.jp

※今回利用するSmartDBの環境を指定

リソースパス： /hibiki/rest/3/binders/13164/documents/1

※今回利用するREST API「文書詳細情報取得」を指定

② データの取得であるため、HTTPメソッドにGETを指定

DreamArts

```
①
$URI = "https://seminar.smartdb.jp/hibiki/rest/3/binders/13164/documents/1"
②
$Method = "GET"
③
$Headers = @{
                "Authorization" = "Bearer sf8jNn4rD-E9hHrDMlcwYwuRQjO_2ra8"
                }
④
Invoke-WebRequest $URI -Method $Method -Headers $Headers -UseBasicParsing -OutFile .¥response_GETDoc.json

⑤
$Response = ((Get-Content .¥response_GETDoc.json -Encoding UTF8) | ConvertFrom-Json)
echo "---実行結果---“
echo (“顧客No：” + $Response.items[4].value)
echo (“会社名：” + $Response.items[6].value)
echo "--------------"
```

③は後ほど取扱います

DreamArts

```
①
$URI = "https://seminar.smartdb.jp/hibiki/rest/3/binders/13164/documents/1"
②
$Method = "GET"
③
$Headers = @{
                "Authorization" = "Bearer sf8jNn4rD-E9hHrDMlcwYwuRQjO_2ra8"
                }
④
Invoke-WebRequest $URI -Method $Method -Headers $Headers -UseBasicParsing -OutFile .¥response_GETDoc.json

⑤
$Response = ((Get-Content .¥response_GETDoc.json -Encoding UTF8) | ConvertFrom-Json)
echo "---実行結果---“
echo (“顧客No：” + $Response.items[4].value)
echo (“会社名：” + $Response.items[6].value)
echo "---------------"
```

解説

⑤ PowerShellのREST APIを実行するためのコマンド
※利用するツールに合わせて、REST APIの実行方法を確認する必要がある。

④ PowerShell画面上で実行結果を確認するためのコマンド

1 URI、HTTPメソッドの指定が必要となる

2 URIは、処理を行う対象のリソースを示す

3 HTTPメソッドは、実施する処理内容を示す

問題

SmartDBの顧客マスタにあるX社のデータを削除するには？

※「削除」は「取得」と同じパターン

1. SmartDBのホスト名、文書を削除するREST APIのリソースパスを用いてURIを指定する。
   また、HTTPメソッドには「POST」を指定する。

2. SmartDBのホスト名、文書を削除するREST APIのリソースパスを用いてURIを指定する。
   また、HTTPメソッドには「DELETE」を指定する。

3. URIを指定する必要はないので、HTTPメソッドに「DELETE」を指定する。

4. URIを指定する必要はないので、HTTPメソッドに「POST」と「DELETE」を指定する。

問題

SmartDBの顧客マスタにあるX社のデータを削除するには？
※「削除」は「取得」と同じパターン

1. SmartDBのホスト名、文書を削除するREST APIのリソースパスを用いてURIを指定する。
   また、HTTPメソッドには「POST」を指定する。

2. **SmartDBのホスト名、文書を削除するREST APIのリソースパスを用いてURIを指定する。**
   また、HTTPメソッドには「DELETE」を指定する。

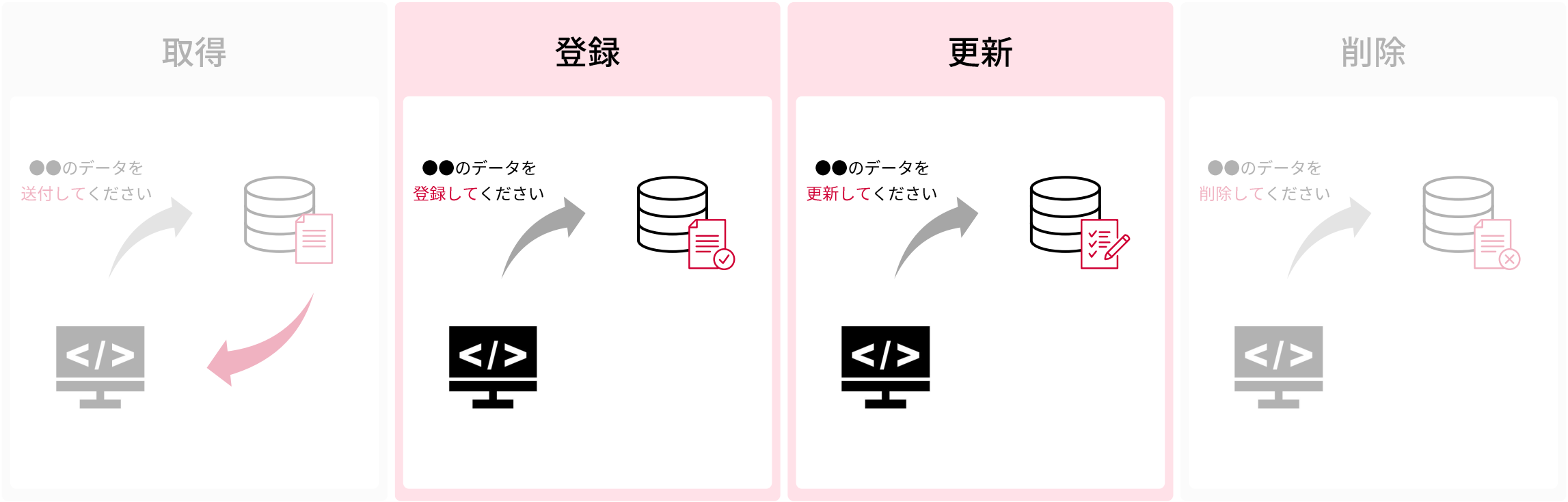
解説

1. REST APIでデータの削除を行う場合、URI、HTTPメソッドを指定する必要がある
2. URIは、①**プロトコル**、②**ホスト名**、③**リソースパス**、④**クエリパラメータ**で構成されており、①、②、③は必ず指定する必要がある
3. HTTPメソッドは、処理に応じた値を指定し、削除の場合は**DELETE**である

登録・更新のリクエスト

登録・更新のリクエスト

DreamArts

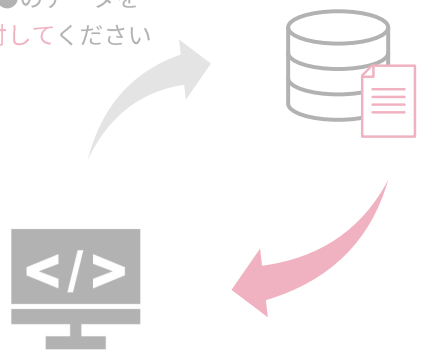


登録・更新のリクエスト

DreamArts

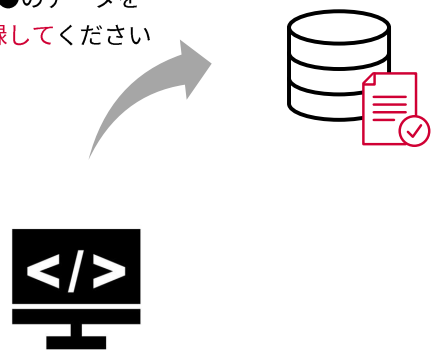
取得

●●のデータを
送付してください



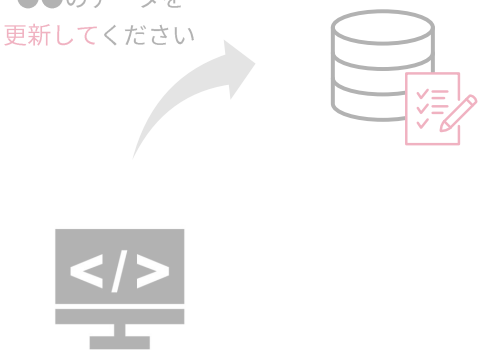
登録

●●のデータを
登録してください



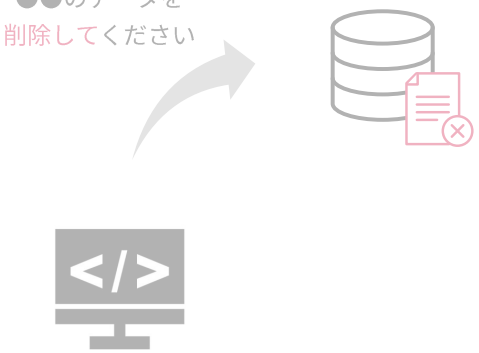
更新

●●のデータを
更新してください



削除

●●のデータを
削除してください



前提

- 営業部に所属するあなたは、Y社を新規顧客として開拓した
- 新規獲得した顧客に関して、全て資料として残す必要がある
- 同部に所属する同僚のBさんは顧客資料の作成も行っている

あなた

同僚 Bさん

あなたは、Bさんにどのような依頼をする？

？

あなた

同僚 Bさん

あなたは、Bさんにどのような依頼をする？

- 顧客資料にY社に情報を追加してください
- Y社の顧客資料を登録してください

あなた

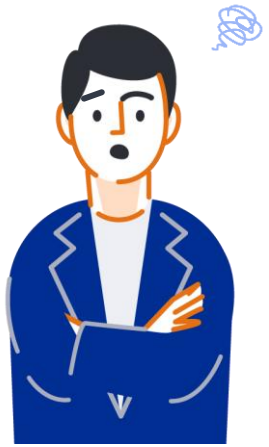
同僚 Bさん

あなたは、Bさんにどのような依頼をする？

- 顧客資料に
- Y社の顧客資

あなた

それだけじゃ資料作れない…



同僚 Bさん

DreamArts

あなたは、Bさんにどのような依頼をする？

？

あなた

同僚 Bさん

あなたは、Bさんにどのような依頼をする？

- 顧客資料にY社に情報を追加してください
- Y社の顧客資料を登録してください

＋

- 住所は・・・で、売上高は・・・です

あなた

同僚 Bさん

あなたは、Bさんにどのような依頼をする？

1. 顧客資料に対して
2. （Y社の）資料を登録してください

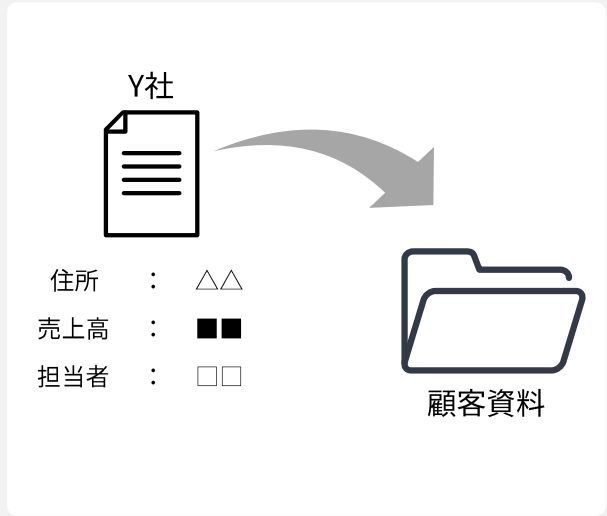
＋

3. 詳細な内容は〜〜〜〜です

あなた

同僚 Bさん

1. ○○というリソースに対して
   顧客資料
2. データを××してください
   登録
3. **データの内容は～～～です**
   住所は・・・
   売上高は・・・



あなた

リクエスト（要求）

同僚 Bさん

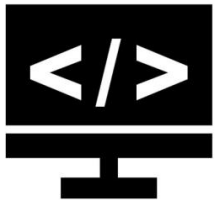
DreamArts

1. 〇〇というリソースに対して
   対象のリソースをURIで指定

2. データを××してください
   実施する処理をHTTPメソッドで指定

＋

3. データの内容は、〜〜〜です
   詳細を**リクエストボディ**で指定



外部プログラム

リクエスト（要求）

Webサービス

DreamArts

JSON形式のリクエストボディを用いてデータの詳細（キーと値）を指定

JSON形式の記述ルール

1. キーと値を「:」で区切る
2. 複数の項目がある場合は、「,」で区切る
3. キーは、必ず「“”」で括る
4. 値が文字列の場合は、「“”」で括る
5. 値が数値の場合は、「“”」で括らない
6. 配列*を入れる場合は、「[ ]」で括る
7. オブジェクト*を入れ子にすることもできる

記述例

```
{
  “Name” : “Y”,
  “Address” : ”東京都〇〇”,
  “Capital” : 100000000,
  “PIC” : [
            “〇田 ×太郎”,
            “□田 △美”
          ],
  “Affiliate1” : {
            “Name” : “Z”,
            “Address” : “大阪府××”
                }
}
```

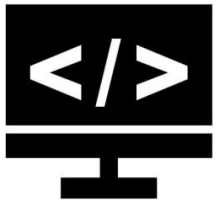
配列

オブジェクト

オブジェクト

【例】SmartDBへのリクエスト 「登録」

1. 顧客情報バインダというリソースに対して
   - https://xxx.smartdb.jp/10001/documents
     バインダID
2. データを登録してください
   - POST
3. 詳細は～～～です
   - {
     “Address” : “東京都渋谷区〇〇”,
     “Sales” : 100000000
     }



外部プログラム

リクエスト（要求）

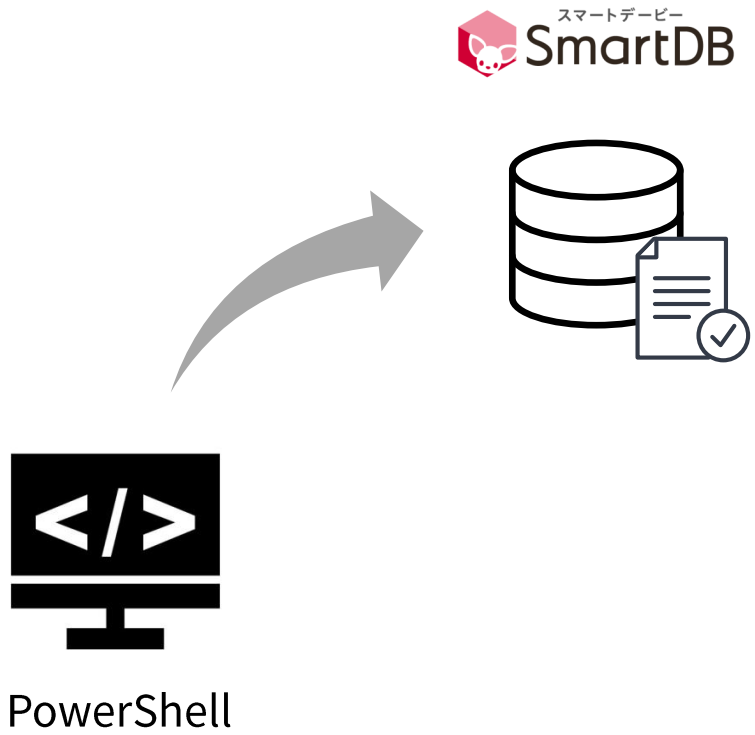
SmartDB

実践：PowerShellでREST APIを実行しよう

顧客マスタに文書を登録

1. 顧客マスタから文書情報を取得しよう

2. 顧客マスタに文書を登録しよう



DreamArts

実行内容の確認

「顧客マスタ」バインダに任意の会社を登録

DreamArts

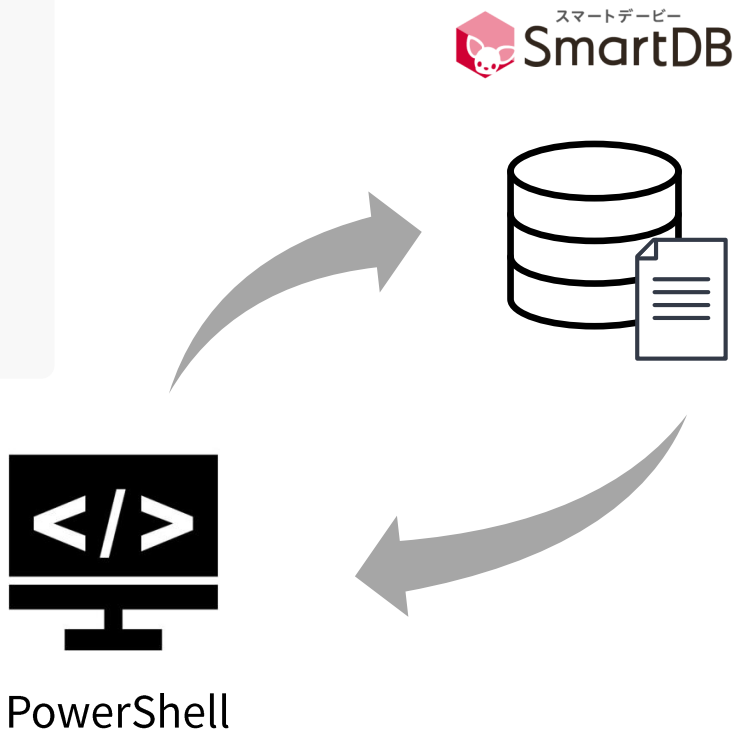
PowerShellでREST APIを実行する方法

1. URI
2. HTTPメソッド
3. リクエストボディ
4. リクエストヘッダ

の指定

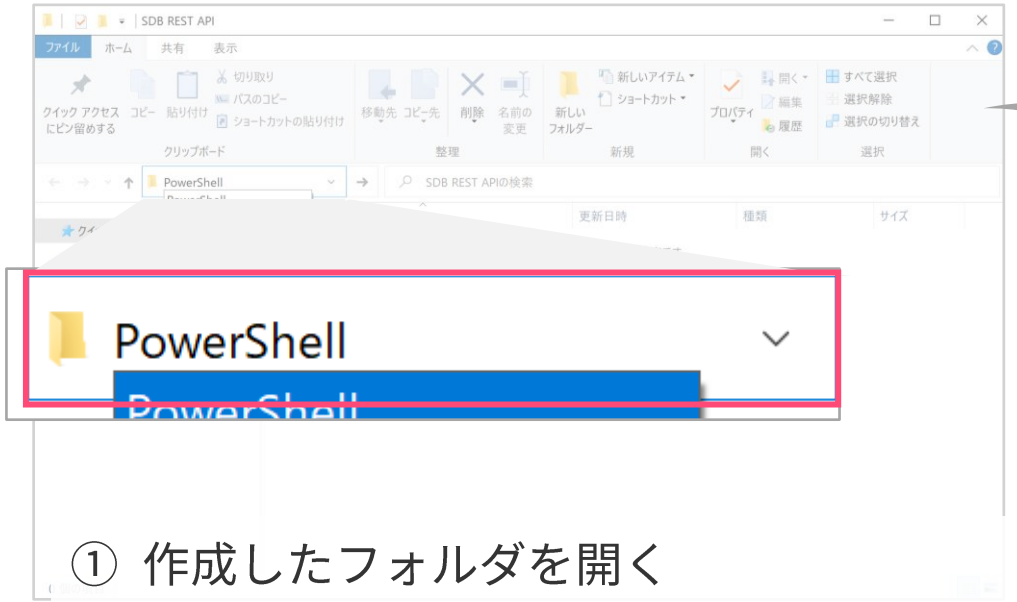
＋

Invoke-WebRequest：REST APIを実行するコマンド

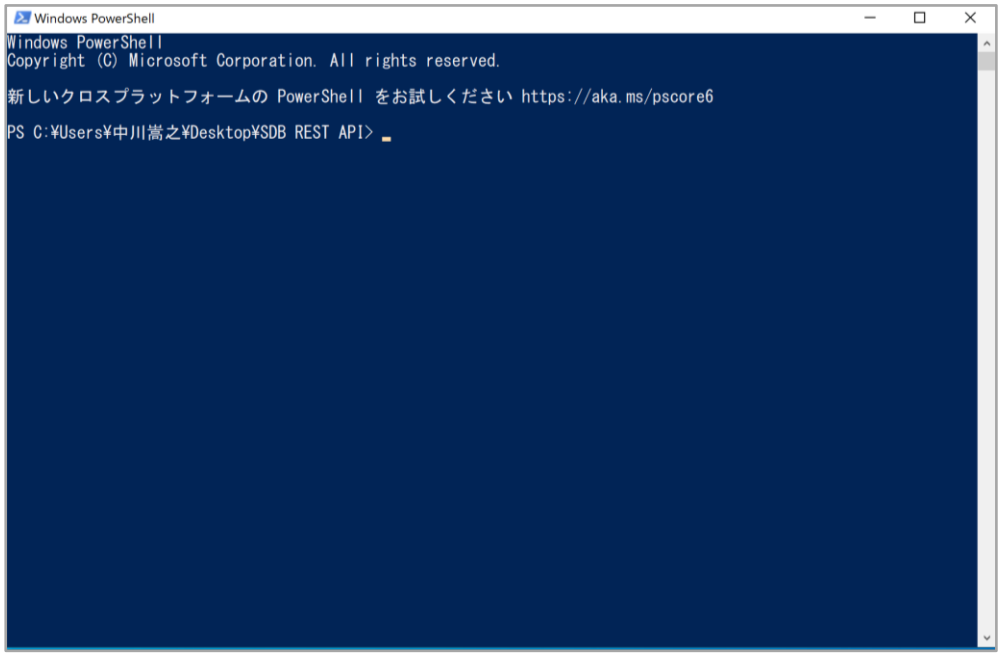


DreamArts

作成したフォルダからPowerShellを起動する



① 作成したフォルダを開く
② アドレスバーに「PowerShell」と入力する
③ Enterキーを押す

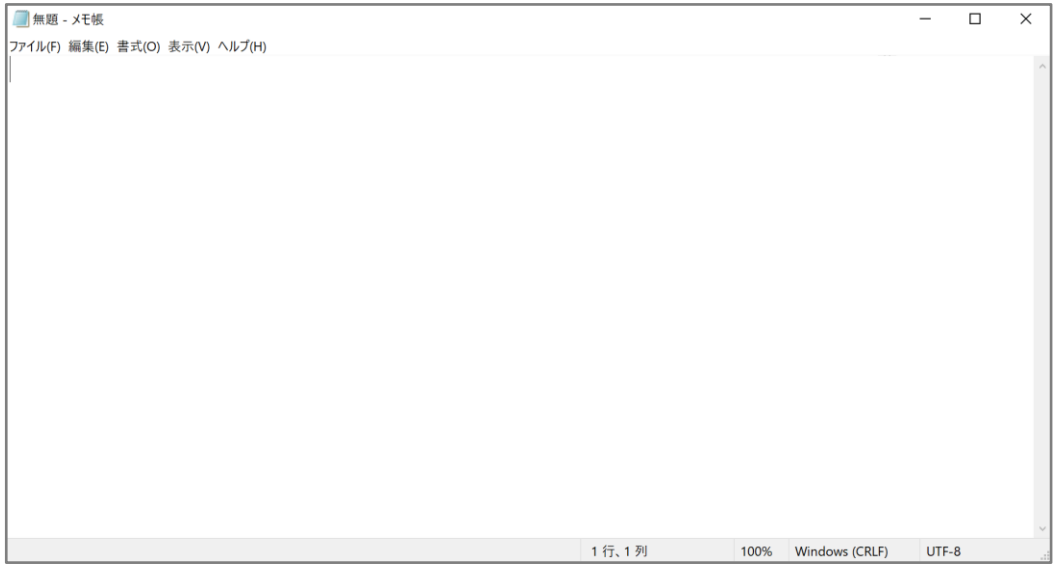


※ログファイルの出力先を今回作成したフォルダにするため

DreamArts

プログラムを作成するためのエディタを起動

① 「メモ帳」を立ち上げる



メモ帳

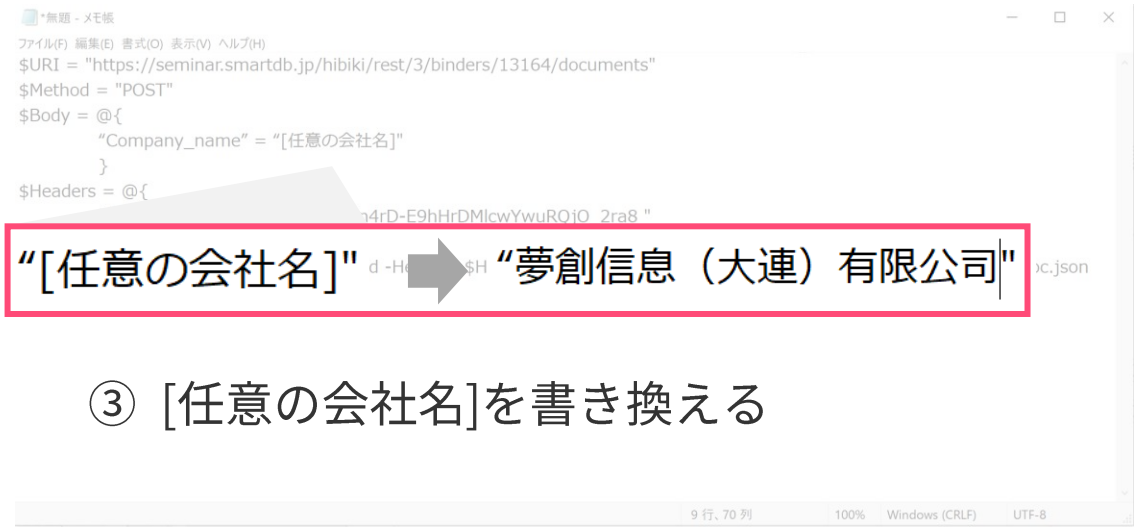
※Vidual Studioなど別のエディタを利用している場合は、そちらでも問題ありません

DreamArts

事前配布のコードをメモ帳にコピペする

① 事前に配布されたPDFの「文書登録」のコードをコピー

② メモ帳に貼り付け



③ [任意の会社名]を書き換える

DreamArts

事前配布のコードをPowerShellにコピペする

① メモ帳で編集したコードをコピー

② PowerShellに貼り付け

③ Enterキーで実行

DreamArts

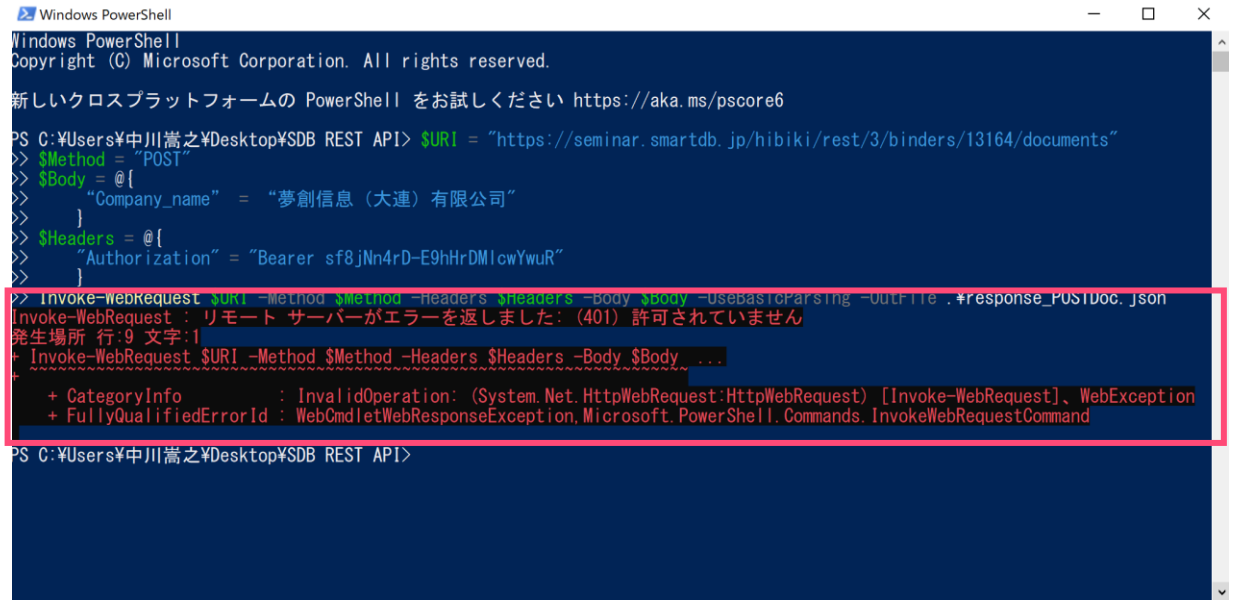
実行結果を確認する

成功

エラーが出力されない

※実行結果を確認するコードを入れていないため結果は出力されません

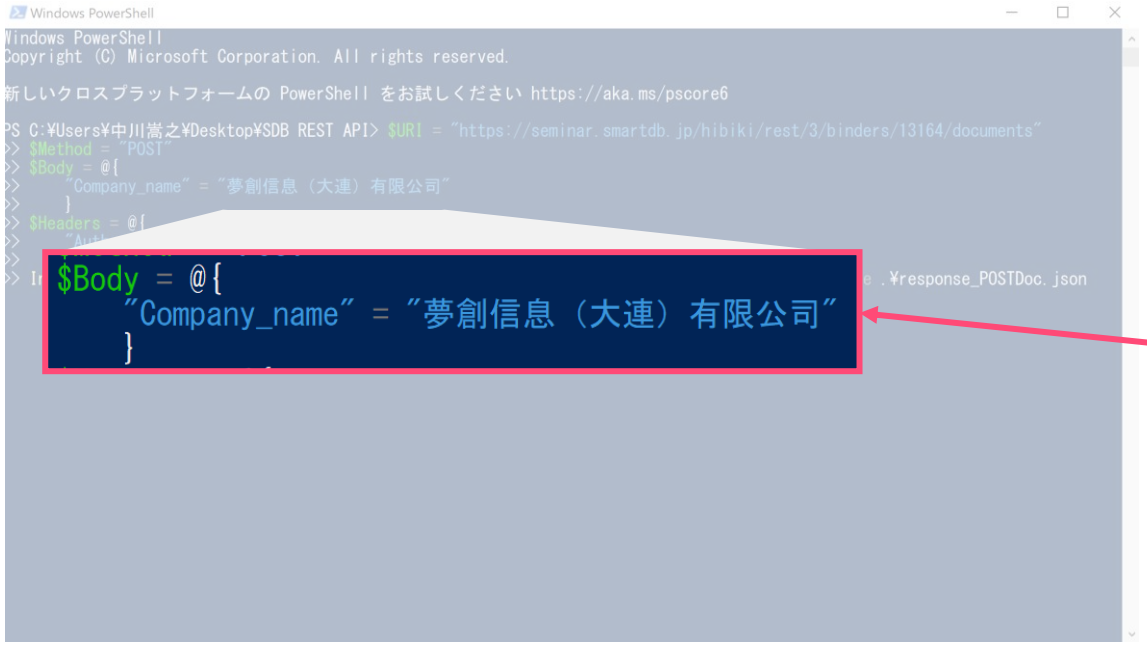
失敗



赤字でエラーが出力

DreamArts

実行結果を確認する



DreamArts

```
①
$URI = "https://seminar.smartdb.jp/hibiki/rest/3/binders/13164/documents"
②
$Method = “POST“
③
$Body = @{
            “Company_name” = “[任意の会社名]"
```

解説

① プロトコル、ホスト名、リソースパスからURIを作成
プロトコル：https
ホスト名： seminar.smartdb.jp
※今回利用するSmartDBの環境を指定
リソースパス： /hibiki/rest/3/binders/13164/documents
※今回利用するREST API「文書新規登録」を指定

② データの取得であるため、HTTPメソッドにPOSTを指定

```
①
$URI = "https://seminar.smartdb.jp/hibiki/rest/3/binders/13164/documents"
②
$Method = “POST“
③
$Body = @{
            “Company_name” = “[任意の会社名]"
            }
④
$Headers = @{
            "Authorization" = "Bearer sf8jNn4rD-E9hHrDMlcwYwuRQjO_2ra8"
            }
```

解説

③ 登録したいデータ詳細のリクエストボディをJSON形式で作成

※PowerShellの場合、「=」を用いて作成しているが、通常は「:」で区切られる。
利用するツールに合わせて、JSON形式のオブジェクトの作成方法を確認する必要がある。

```
①
$URI = "https://seminar.smartdb.jp/hibiki/rest/3/binders/13164/documents"
②
$Method = “POST“
③
$Body = @{
            “Company_name” = “[任意の会社名]"
            }
④
$Headers = @{
            "Authorization" = "Bearer sf8jNn4rD-E9hHrDMlcwYwuRQjO_2ra8"
            }
⑤
Invoke-WebRequest $URI -Method $Method -Headers $Headers -UseBasicParsing -OutFile .¥response_GETDoc.json
```

④は後ほど取扱います

DreamArts

```
①
$URI = "https://seminar.smartdb.jp/hibiki/rest/3/binders/13164/documents"
②
$Method = “POST“
③
$Body = @{
                “Company_name” = “[任意の会社名]"
                }
④
$Headers = @{
                "Authorization" = "Bearer sf8jNn4rD-E9hHrDMlcwYwuRQjO_2ra8"
                }
⑤
Invoke-WebRequest $URI -Method $Method -Headers $Headers -UseBasicParsing -OutFile .¥response_GETDoc.json
```

解説

⑤ PowerShellのREST APIを実行するためのコマンド

※利用するツールに合わせて、REST APIの実行方法を確認する必要がある。

1 URI、HTTPメソッド加え、**リクエストボディ**の指定が必要となる

2 リクエストボディは、主にJSON形式で**データの詳細**を示す

3 JSON形式とは、**データの記述方式**の1つである

問題

SmartDBの顧客マスタにあるY社のデータを更新するには？

※「更新」は「登録」と同じパターン

1. HTTPメソッドには「PUT」を指定、リクエストボディを指定する必要はない。
2. HTTPメソッドには「PUT」を指定、 JSON形式でリクエストボディを指定する。
3. HTTPメソッドには「POST」を指定し、リクエストボディを指定する必要はない。
4. HTTPメソッドには「POST」を指定、JSON形式でリクエストボディを指定する。

問題

SmartDBの顧客マスタにあるY社のデータを更新するには？
※「更新」は「登録」と同じパターン

1. HTTPメソッドには「PUT」を指定、リクエストボディを指定する必要はない。

2. **HTTPメソッドには「PUT」を指定、 JSON形式でリクエストボディを指定する。**

解説

1. REST APIでデータの更新を行う場合、URI、HTTPメソッドに加えて**リクエストボディ**を指定する必要がある
2. HTTPメソッドは、処理に応じた値を指定し、更新の場合は**PUT**である
3. リクエストボディは主に**JSON形式**というデータ形式で記述される

補足情報が必要なリクエスト

前提

- 営業部に所属するあなたは、アメリカ企業のα社の情報が必要となった
- NY支社に勤めるCさんがα社の情報を持っている
- あなたは、Cさんとの面識がない

あなた

NY支社 Cさん

あなたは、Cさんにどのような依頼をする？

- α社に関する情報をください
- α社の顧客資料を送ってください

あなた

海外支社 Cさん

あなたは、Cさんにどのような依頼をする？

- α社に
- α社の顧

この人は誰…？
機密情報は教えられないよ…

あなた

海外支社 Cさん

あなたは、Cさんにどのような依頼をする？

？

あなた

海外支社 Cさん

あなたは、Cさんにどのような依頼をする？

- α社の顧客資料を送ってください

＋

- 私は本社の～～です

あなた

海外支社 Cさん

あなたは、Cさんにどのような依頼をする？

1. 顧客資料に対して
2. （α社の）資料を取得してください
3. 詳細な内容は〜〜〜〜です

＋

4. 私は本社の〜〜です、フォーマットは××にしてください

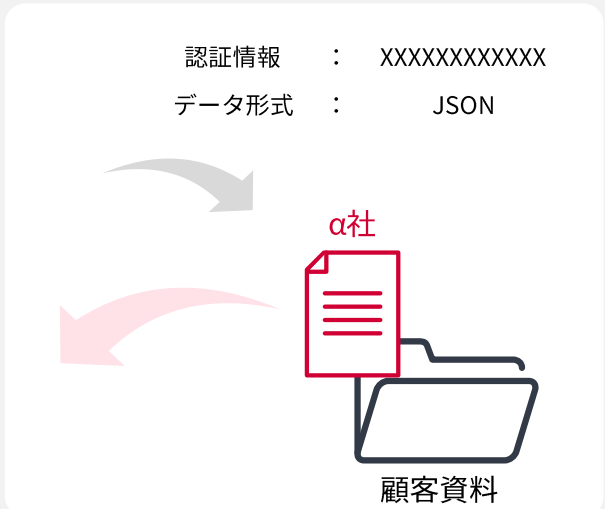
※「取得」のため、3.の指定は不要

あなた

海外支社 Cさん

1. ○○というリソースに対して
   顧客資料のうちのα社の資料
2. データを××してください
   取得
3. データの内容は～～～です
4. **補足情報は～～～です**
   私は・・・
   フォーマットは・・・

あなた



※「取得」のため、3.の指定は不要

海外支社 Cさん

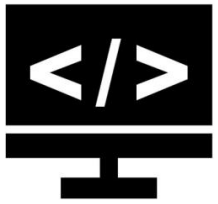
1.〇〇というリソースに対して
対象のリソースをURIで指定

2.データを××してください
実施する処理をHTTPメソッドで指定

3.データの内容は〜〜〜です
詳細をリクエストボディで指定

＋

4. 補足情報は〜〜〜です
認証などの補足情報を**リクエストヘッダ**で指定



外部プログラム

リクエスト（要求）

Webサービス

DreamArts

Key-Value形式でリクエストの形式、認証情報を指定

Key-Value形式の記述ルール

1. キーと値を 「:」 で区切る
2. 値にスペースを含む場合は、「“”」で括る
3. 複数の項目がある場合は、改行で区切る

記述例

① Authrization : Bearer XXXXXXXXXXXXX
② Content-Type : application/json
③ Accept : application/json

ヘッダー　値

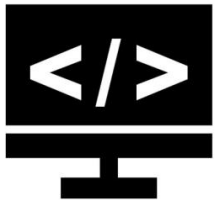
① 認証情報をWebサービスに伝える
② リクエストのデータ形式・種類をWebサービスに伝える
③ レスポンスのデータ形式・種類の要望をWebサービスに伝える

1.顧客情報バインダのα社の文書というリソースに対して
- https://xxx.smartdb.jp/10001/documents/10
  バインダID　文書ID

2.データを取得してください
- GET

3.詳細は～～～です
- ※GETのため不要

4.認証情報はXXXXです、JSON形式でデータをください
- Authorization : Bearer XXXX
  Accept : application/json



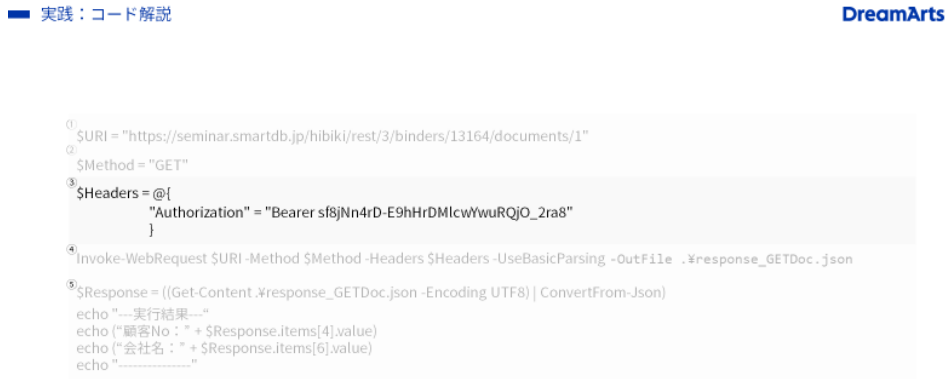
外部プログラム

リクエスト（要求）

SmartDB

実践：PowerShellでREST APIを実行しよう

補足情報の書き方




```
$Headers = @{
            "Authorization" = "Bearer sf8jNn4rD-E9hHrDMlcwYwuRQjO_2ra8"
            }
```

解説

SmartDBではREST APIの実行に認証が必要なため、リクエストヘッダで認証情報を作成

※SmartDBの場合、Bearerトークンという方式を採用している。
他のWebサービスでは異なる認証方式を採用している場合がある。

1. **リクエストヘッダ**の指定が必要となる
2. リクエストヘッダは、**認証**や**データ形式**などの補足情報を示す
3. リクエストヘッダは、**Key-Value形式**で記述される

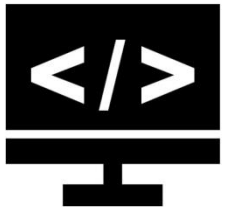
DreamArts

1. REST APIとは？
2. REST APIの利用方法と実践
   - リクエストを作る
   - レスポンスを読む
3. まとめ
4. 次回予告

DreamArts

Webサービスからレスポンスが返却される

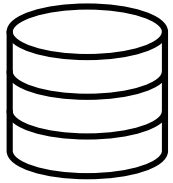
顧客マスタにある顧客IDがA00001の
顧客名を教えてください！



外部プログラム

リクエスト（要求）

レスポンス（応答）



Webサービス

了解です！
顧客名は株式会社ドリーム・アーツです。

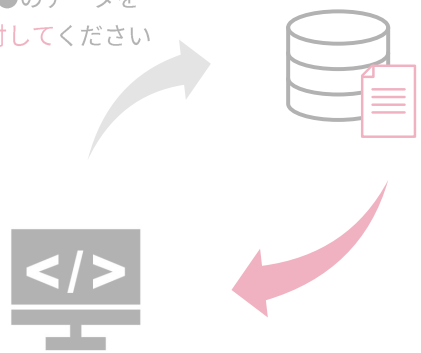
登録・更新・削除のレスポンス

登録・更新・削除のレスポンス

DreamArts

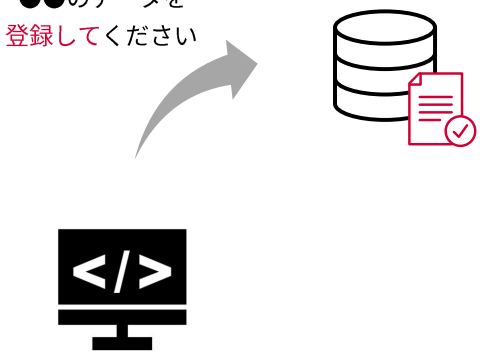
取得

●●のデータを
送付してください



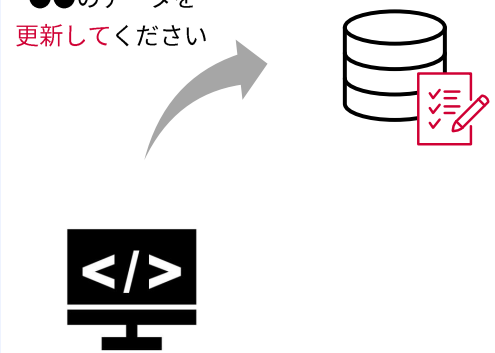
登録

●●のデータを
登録してください



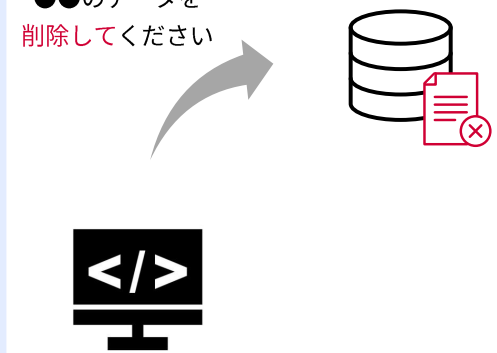
更新

●●のデータを
更新してください



削除

●●のデータを
削除してください

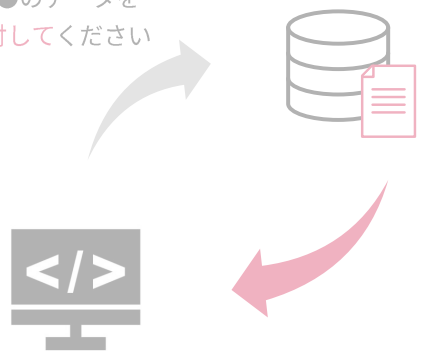


登録・更新・削除のレスポンス

DreamArts

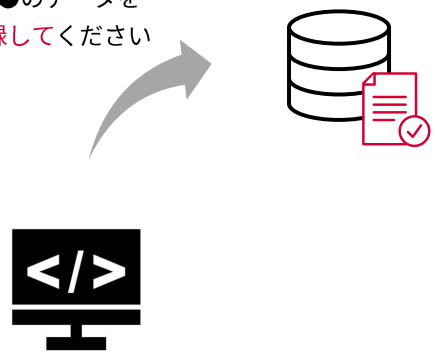
取得

●●のデータを
送付してください



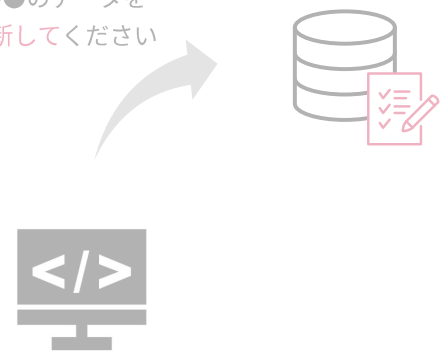
登録

●●のデータを
登録してください



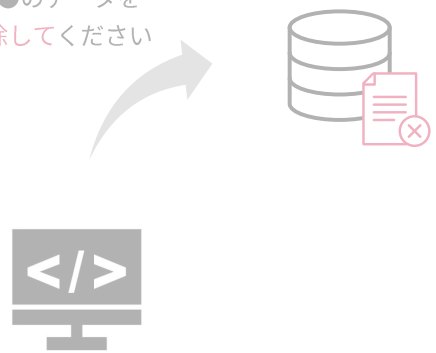
更新

●●のデータを
更新してください



削除

●●のデータを
削除してください



Bさんはどのような報告をする？

- 顧客資料にY社に情報を追加してください
- Y社の顧客資料を登録してください
- 住所は××で、売上高は〇〇で、・・・です

あなた

同僚 Bさん

Bさんはどのような報告をする？

？

あなた

同僚 Bさん

DreamArts

Bさんはどのような報告をする？

- 了解です、登録しました
- 完了しました
- すみません、××な理由でできませんでした

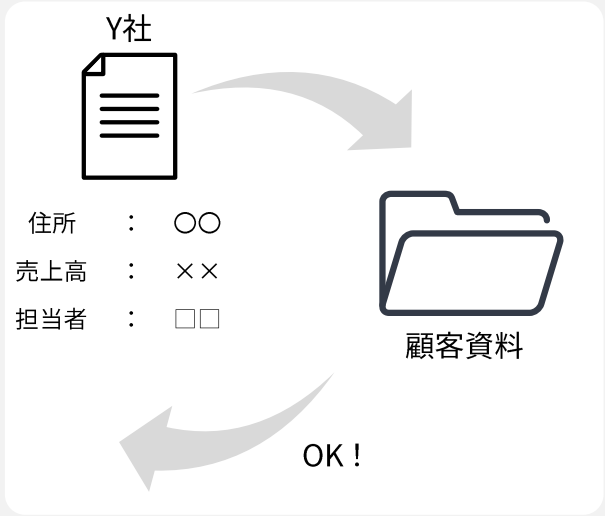
あなた

同僚 Bさん

DreamArts

1. リクエストを～～～しました

正常に処理しました
〇〇な理由で処理に失敗しました



あなた

レスポンス（応答）

同僚 Bさん

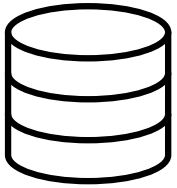
DreamArts

1. リクエストを〜〜〜しました

実行結果を**ステータスコード**で返却

外部プログラム

レスポンス（応答）



Webサービス

DreamArts

3桁のステータスコードで実行結果を返却

5つのパターン

100番台　：　リクエストが処理中

200番台　：　リクエストを正常に処理

300番台　：　リダイレクトが発生

400番台　：　主にクライアント側のエラーが発生

500番台　：　主にサーバー側のエラーが発生

返却例

```
①"code": 400
②"message": "error message"
```

① ステータスコード
② エラー等を伝えるメッセージ

DreamArts

3桁のステータスコードで実行結果を返却

5つのパターン

100番台　：　リクエストが処理中

成功 200番台　：　リクエストを正常に処理

300番台　：　リダイレクトが発生

失敗 400番台　：　主にクライアント側のエラーが発生

失敗 500番台　：　主にサーバー側のエラーが発生

返却例

```
①"code": 400,
②"message": "error message"
```

① ステータスコード
② エラー等を伝えるメッセージ

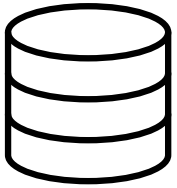
【例】SmartDBからのレスポンス 「登録」

1.リクエストを正常に処理しました

▶ 200

外部プログラム

レスポンス（応答）



SmartDB

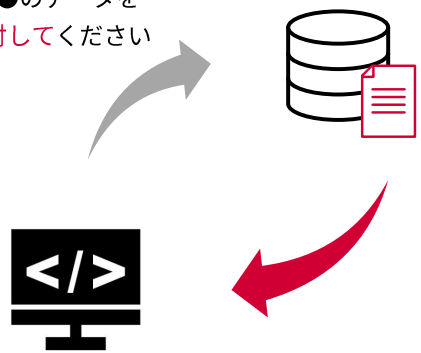
取得のレスポンス

取得のレスポンス

DreamArts

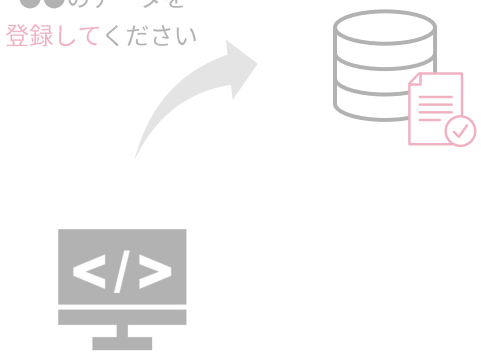
取得

●●のデータを
送付してください



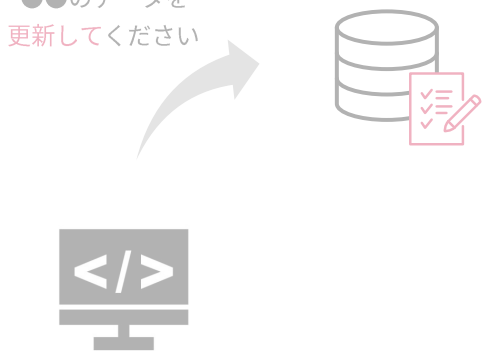
登録

●●のデータを
登録してください



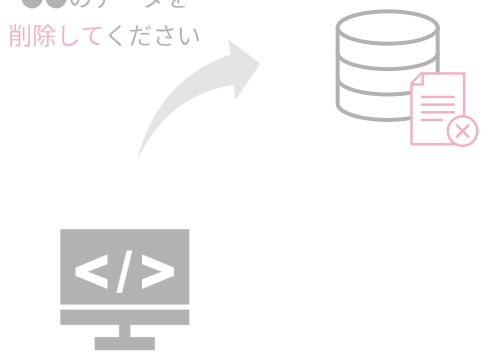
更新

●●のデータを
更新してください



削除

●●のデータを
削除してください



あなたは、Bさんにどのような依頼をする？

- 顧客資料の中にあるX社に関する情報をください
- X社の顧客資料を見せてください

あなた

同僚 Bさん

Bさんはどのような報告をする？

？

あなた

同僚 Bさん

DreamArts

Bさんはどのような報告をする？

- 了解です、登録しました
- 完了しました
- すみません、××な理由でできませんでした

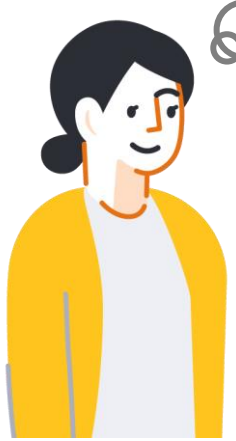
あなた

同僚 Bさん

DreamArts

Bさんはどのような報告をする？

X社の詳細は・・・？



あなた

でできませんでした

同僚 Bさん

DreamArts

Bさんはどのような報告をする？

- 了解です

＋

- 住所は・・・で、売上高は・・・です

あなた

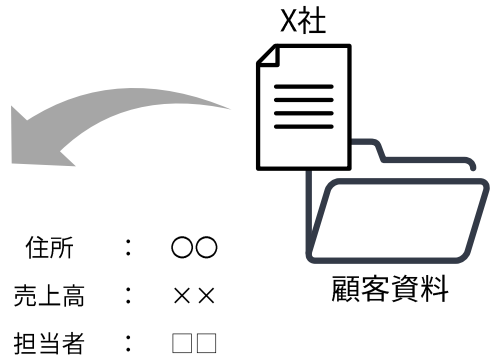
同僚 Bさん

1. リクエストを～～～しました

   正常に処理しました
   ○○な理由で処理に失敗しました

2. **データの内容は～～～です**

   住所は・・・
   売上高は・・・



あなた

レスポンス（応答）

同僚 Bさん

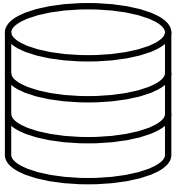
1. リクエストを〜〜〜しました
   実行結果をステータスコードで返却

＋

3. データの内容は、〜〜〜です
   詳細を**レスポンスボディ**で返却

外部プログラム

レスポンス（応答）



Webサービス

DreamArts

JSON形式のレスポンスボディでデータの詳細（キーと値）を返却

階層の参照方法

1. キーによって参照
2. オブジェクトが入れ子となっている場合、
   キーを「.」で繋いで表現
3. 配列がある場合、キーと「[番号]」で表現
4. 「[番号]」は0からスタート

返却例

```
{
  “Name” : “Y”,
  “Address” : ”東京都〇〇”,
  “Capital” : 100000000,
  “PIC” : [
            “〇田 ×太郎”,
            “□田 △美”
          ],
  “Affiliate1” : {
            “Name” : “Z”,
            “Address” : “大阪府××”
                }
}
```

配列

オブジェクト

オブジェクト

DreamArts

JSON形式のレスポンスボディでデータの詳細（キーと値）を返却

返却例

CompanyObject =
オブジェクトに名前をつける

```
{
  “Name” : “Y”,
  “Address” : ”東京都〇〇”,
  “Capital” : 100000000,
  “PIC” : [
            “〇田 ×太郎”,
            “□田 △美”
          ],
  “Affiliate1” : {
            “Name” : “Z”,
            “Address” : “大阪府××”
                }
}
```

参照例

Y	CompanyObject.Name
大阪府××	CompanyObject.Affiliate1.Address
〇田 ×太郎	CompanyObject.PIC[0]

取得したい値　　参照例

1.リクエストを正常に処理しました

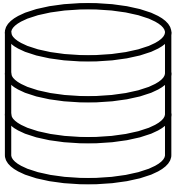
- 200

2.詳細は～～～です

- {
  “Address” : “東京都渋谷区〇〇”,
  “Sales” : 100000000
  }

外部プログラム

レスポンス（応答）



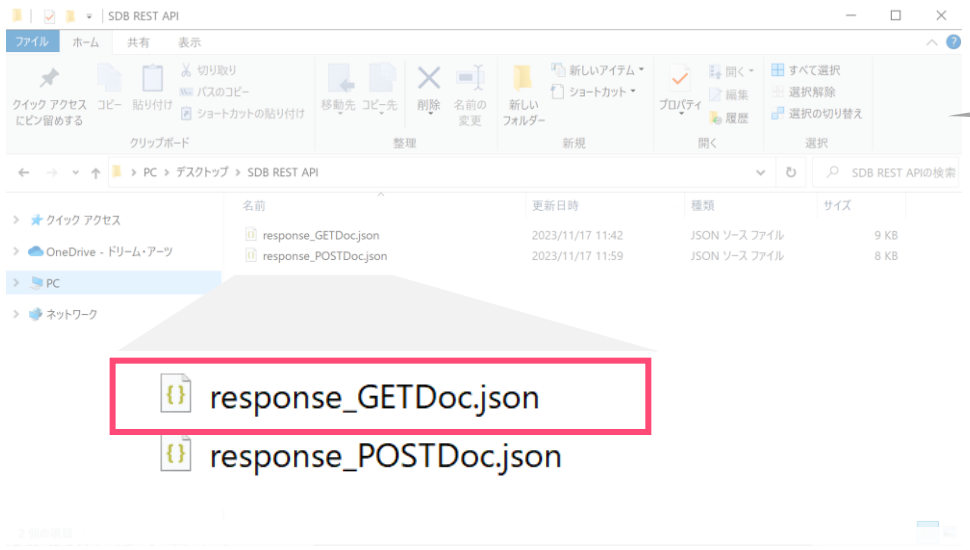
SmartDB

実践：PowerShellでREST APIを実行しよう

レスポンスの読解

DreamArts

保存されたJSONファイルを開く



① response_GETDoc.jsonを開く

response_GETDoc.json

C: > Users > 中川嵩之 > Desktop > SDB REST API > {} response_GETDoc.json > ...

```
{
    "created": {
        "key": "t_nakagawa@dreamarts.co.jp",
        "name": "中川 嵩之",
        "primary": 2000000,
        "primaryGroupName": "株式会社ドリーム・アーツ",
        "email": "t_nakagawa@dreamarts.co.jp",
        "disabled": false,
        "status": 0,
        "lang": {
            "ja": {
                "name": "中川 嵩之",
                "first_name": "嵩之",
                "last_name": "中川",
                "primary_gname": "株式会社ドリーム・アーツ"
            },
            "en": {
                "primary_gname": "株式会社ドリーム・アーツ"
            },
            "zh": {
                "primary_gname": "株式会社ドリーム・アーツ"
            }
        },
        "recognizeInfo": "株式会社ドリーム・アーツ",
        "external": false,
        "type": 1,
        "id": 1000193
    },
```

※読みやすくするために、Visual Stadioというエディタで読み込んでいます

DreamArts

実行結果を確認する

左図のデータは、右図の文書データが
JSON形式に変換されたものである

DreamArts

1 登録、更新、削除では、結果を示す**ステータスコード**が返却される

2 取得では、データの詳細を示す**レスポンスボディ**も返却される

3 レスポンスボディは、**JSON形式**で返却される

問題 レスポンスの説明として正しいのは？

1. ステータスコードが200の場合は正常に処理が行われており、GETの場合はデータの詳細がレスポンスボディで返却される。

2. ステータスコードが400の場合は正常に処理が行われており、GETの場合はデータの詳細がレスポンスボディで返却される。

3. ステータスコードが200の場合はエラーが発生しており、レスポンスでデータの詳細を返却することはない。

4. ステータスコードが400の場合はエラーが発生しており、レスポンスでデータの詳細を返却することはない。

DreamArts

問題 レスポンスの説明として正しいのは？

1. **ステータスコードが200の場合は正常に処理が行われており、GETの場合はデータの詳細がレスポンスボディで返却される。**

解説

1. レスポンスでは、**ステータスコード**、**レスポンスボディ**が返却される
2. ステータスコードは、処理が正常に行われた場合に**200番台**が返却される
3. レスポンスボディは、主に**JSON形式**というデータ形式で返却される

細を返却することはない。

4. ステータスコードが400の場合はエラーが発生しており、レスポンスでデータの詳細を返却することはない。

DreamArts

1. REST APIとは？
2. REST APIの利用方法と実践
3. まとめ
4. 次回予告

1. APIとは、外部から**機能・情報を利用するための窓口**である
2. APIには**リクエスト**と**レスポンス**という2つの通信がある
3. REST APIでは、データの**取得**、**登録**、**更新**、**削除**ができる
4. リクエストは、主に4つの要素で構成される

要素	説明
URI	：処理を行う対象のリソース
HTTPメソッド	：実行する処理内容
リクエストボディ	：登録、更新するデータの詳細
リクエストヘッダ	：認証などの補足情報

5. レスポンスは、主に2つの要素で構成される

要素	説明
ステータスコード	：実行した結果
レスポンスボディ	：返却されたデータの詳細

DreamArts

1. REST APIとは？
2. REST APIの利用方法と実践
3. まとめ
4. 次回予告

次回予告

No.	実施概要	詳細内容	推奨者
本日	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
次回	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
3	SmartDB REST API アプリ操作実践編	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
4	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

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

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

フォローする

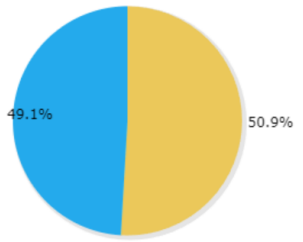
のほほん Great supporter Nice multiple posts

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%

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

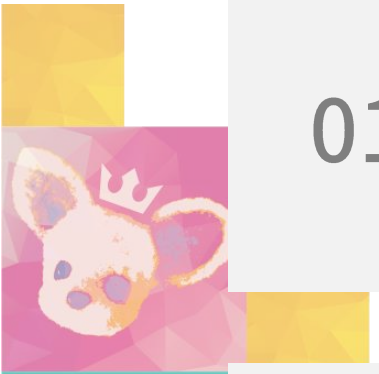
SmartDB Certified Specialist

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/

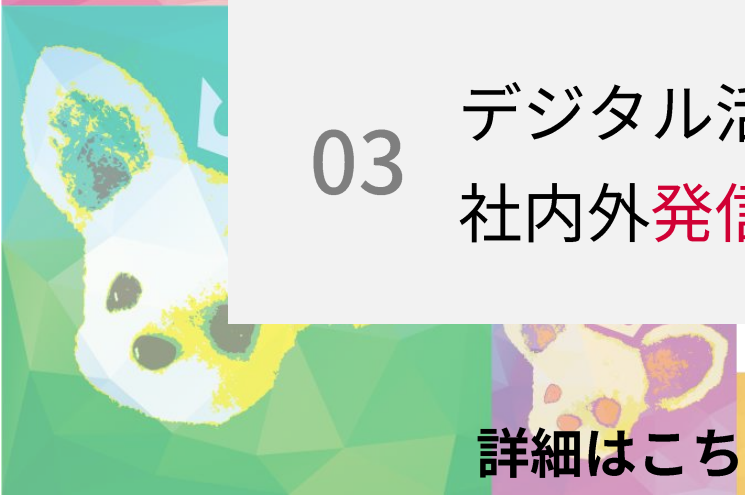
DreamArts

認定制度を活用することで…



01 スキルの可視化

02 業務デザイナー育成



03 デジタル活用の社内外発信

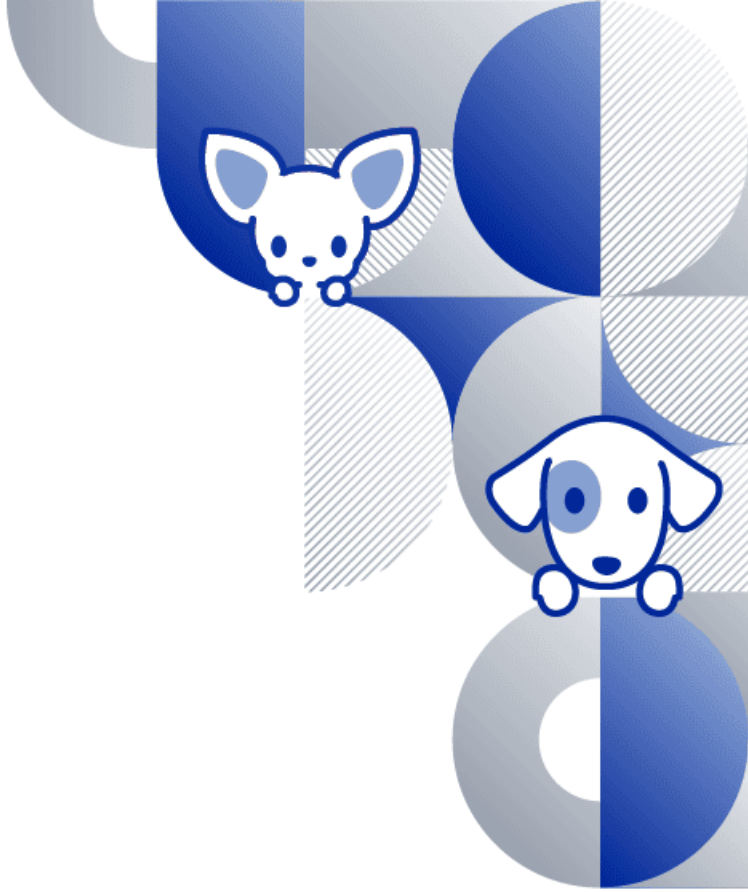
04 DX企業文化への変革

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

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



DreamArts

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