Power Platform

French Summit

BIZ APPS FRENCH COMMUNITY

aMP

Club Power BI

avec le soutien de Microsoft

Merci à nos sponsors!

Power Platform
French Summit

SUIVEZ-NOUS !

Power Platform
French Summit

BIZ APPS
FRENCH COMMUNITY

https://www.linkedin.com/company/biz-apps-french-community

@BizFrench

aMP

https://www.linkedin.com/company/communaut-aos

@aOSComm

Club Power BI

https://www.linkedin.com/groups/8599174/

@ClubPowerBI

Power Platform
French Summit

Power Apps Canvas :

Les fonctions définies par l'utilisateur 🔥 ⚡ 🚀

Tanguy BADIER

Power Platform Tech Leader
Governance & adoption
specialist

devoteam

David ZOONEKYNDT

Power Platform Expert
Microsoft MVP

devoteam

AGENDA

- Principe
- Définir une fonction
- Types de données
- Topic 4

Principe

Principe - Généralités

- Les UDFs permettent de créer nos propres fonctions en langage Power Fx
- Elles sont déclarées dans la propriété **Formulas** de l'objet **App**
- Elles s'utilisent exactement de la même manière que les formules Power Fx :

NomFormule*(*'Param. 1', 'Param. 2', ...*) = *Résultat

Principe - Propriétés

Les UDFs partagent les mêmes propriétés que les **formules nommées** :

- La définition de la fonction est immuable
- Le résultat est toujours à jour

Principe – Code vs Low Code

Power Platform
French Summit

```
Class CalculParPondération () {

    const int n0 = 3;
    const int raison = 3 ;

    function int CalculPointPondération( int NbPersonne, int
Classement, int PointRépartitionGlobal) {
        return PointRépartitionGlobal * CalculDesPoids(NbPersonne,
Classement) / CalculdesPoidsTotal(NbJoueur)
    }

    function int CalculDesPoids( int NbPersonne, int Classement) {
        return pas * ( NbPersonne - Classement) + raison ;
    }

    function int CalculdesPoidsTotal(int NbPersonne) {

        return NbPersonne * (( n0 + NbJoueur * raison ) /2 );
    }

}
```

```
n0 = 3;;

raison= 3;;

udfCalculPointPondération(NbPersonne: Number; Classement:
Number; NbPointRepartition:Number):Number =
NbPointRepartition*udfCalculdesPoids(NbPersonne;Classement)/
udfCalculPoidsTotal(NbPersonne);;

udfCalculdesPoids(NbPersonne:Number;
Classement:Number):Number = raison *(NbPersonne-
Classement)+n0;;

udfCalculPoidsTotal(NbPersonne:Number):Number = NbPersonne*
((n0+NbPersonne*raison)/2);;
```

Principe - Avantages

- Le résultat est toujours disponible
- Le calcul de la fonction peut être différé
- Moins de code à produire
- Moins de code à maintenir

Principe - Limitations

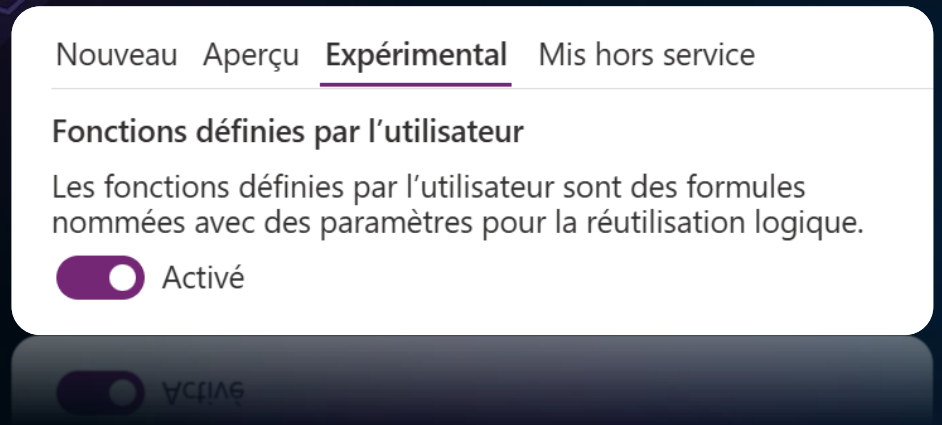
Elles ont cependant quelques **limitations** :

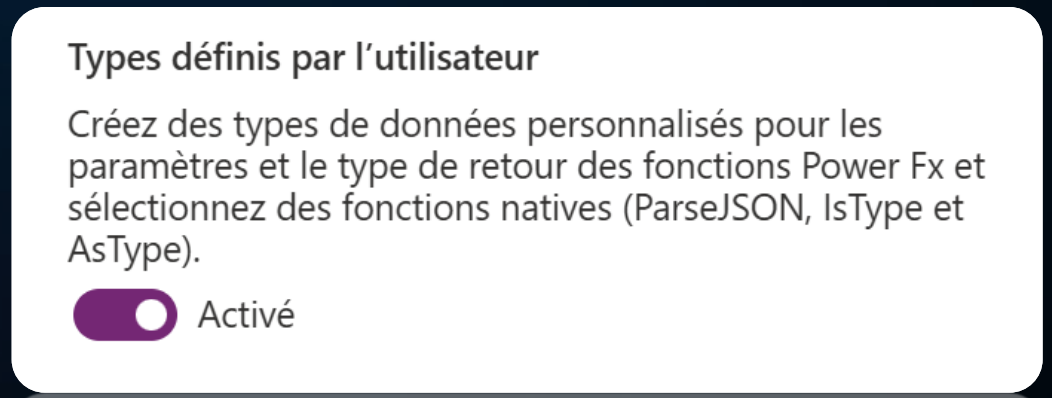
- Pas de récursivité, ni de séquentiel
- Pas d'appel de formules comportementales
- Les paramètres d'entrée définis ne peuvent pas être optionnels
- Ne pas appeler les paramètres d'un objet ou d'un composant

Principe - Prérequis

Power Platform
French Summit

Pour bénéficier pleinement des UDF, deux paramètres à activer :





Définir une fonction

Définir une fonction

Dans une UDF, on devra définir :

1. Le nom de la fonction
2. Les paramètres d'entrée et leur format
3. Le format de sortie
4. Les calculs à opérer

DÉFINIR UNE FONCTION

Exemple de syntaxe :

```
// Calcul simple de deux nombres
udfPoucentage( Actuel: Number ; Maxi: Number ): Number= IfError(Actuel/Maxi*100;Blank());;
```

1. Nom de la fonction
2. Paramètres d'entrée et format
3. Format de sortie
4. Calcul de la sortie

DÉFINIR UNE FONCTION

Exemple de syntaxe :

On termine la formule !

5

Types de données

Types de données

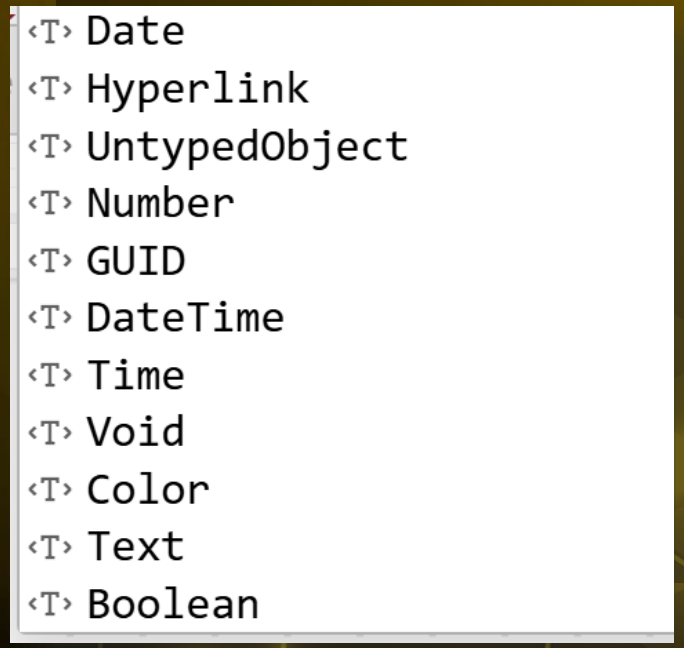
Dans une UDF, le format doit être spécifié par un type de données pour :

- Les paramètres d'entrée
- Le résultat de la sortie

Types de données

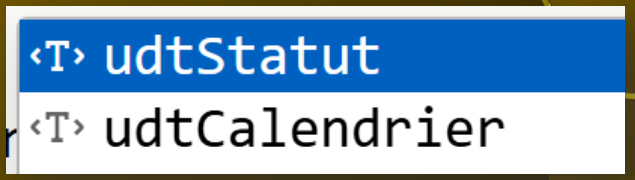
Les types de données peuvent être :

DataTypes de base



DataTypes d'une source

Défini par l'utilisateur



Types de données

Issus d'une source :

Dès qu'une source de donnée est ajoutée à l'application, son DataType est disponible.

Attention, pour l'instant seul le **format table** est proposé, pas le format pour un enregistrement

Types de données

Défini par l'utilisateur :

La définition d'un type de données personnalisée se fait également dans l'objet App.Formulas

Types de données

On définit un type de données personnalisé avec le symbole :=

Une fois défini, il peut être utilisée dans une fonction

```
// Type enregistrement
udtStatut:=Type({

    Code:Number;
    Nom:Text;
    Couleur:
    Color;
    EnFonction:Boolean
    });;
```

```
// Changement de couleur en fonction de la valeur du Statut; si le statut est inconnu; on renvoie du gris
udfStatutColor(StatutCode:Number):udtStatut=
Coalesce(LookUp(nfStatuts;Code=StatutCode);{Code:0;Nom:"Iconnu";Couleur:ColorValue("#999999");EnFonction: false });;
```

Types de données

Mais aussi pour définir d'autres types de données !

Par exemple, quand je définis le type de données pour un enregistrement:

```
udtCalJour:=
Type({Date:Date;Détail:Text;IsWE:Boolean;IsDim:Boolean;IsHoliday:Boolean});;
```

Je peux le réutiliser pour définir une table sur le même modèle :

```
udtCalendrier:=Type([udtCalJour]);;
```

UNE PETITE DÉMO ?

TÉLÉCHARGER L'APPLICATION DÉMO

https://github.com/DavidZoon/PowerApps_UDF_Demo

Power Platform
French Summit

Merci pour votre attention!

La discussion se poursuit dans l'application run.events 😉

BIZ APPS FRENCH COMMUNITY

aMP

Club Power BI

avec le soutien de Microsoft