COGNOS®

Cognos Impromptu (R)

Manuel de référence de PowerPrompts

Informations sur le produit

Le présent document s'applique à Impromptu (R) Version 7.1 et peut aussi s'appliquer aux versions ultérieures de ce produit. Pour vérifier s'il existe une version plus récente du document, visitez le site Web du support client de Cognos (http://support.cognos.com).

Copyright

Table des matières

Chapitre 1 : Création d'une application PowerPrompts

L'application PowerPrompts guide les usagers de rapports, via un ensemble de pages HTML à partir desquelles ils peuvent sélectionner les informations qu'ils souhaitent inclure dans un rapport. L'usager du rapport voit alors le rapport généré s'afficher, en fonction des sélections effectuées. Par exemple, grâce à l'application PowerPrompts, les usagers de rapports peuvent ajouter :

- des colonnes à un rapport,
- une mise en forme aux colonnes,
- un modèle de rapport,
- une mise en forme conditionnelle,
- des filtres.

Utilisez PowerPrompts Developer Studio pour créer ces applications basées sur le Web pour Impromptu Web Reports.

PowerPrompts est disponible avec la version administrateur d'Impromptu.

Utilisez la liste de contrôle suivante lors de la création d'une application PowerPrompts :

- Créez un ensemble de pages HTML qui inviteront les usagers de rapports à définir le contenu d'un rapport.
- Créez les instructions de navigation dans les pages HTML.
- Les pages peuvent inclure des liens de navigation conditionnels afin que les usagers de rapports puissent suivre les branches de l'application PowerPrompts plutôt que d'afficher toutes les pages de l'application.
- Créez des dynamos qui renvoient des données d'une base de données logique, puis ajoutez ces dynamos aux pages HTML appropriées.
- Créez le script permettant de modifier le rapport associé à l'application PowerPrompts.
- Enregistrez les instructions sous forme de fichier d'application PowerPrompts (.xxm).
- Sélectionnez un rapport Impromptu avec lequel tester l'application PowerPrompts.
- Testez l'application PowerPrompts et corrigez les erreurs éventuelles.
- Informez l'administrateur d'Impromptu Web Reports que l'application PowerPrompts est prête à être déployée.

PowerPrompts vous permet de diminuer le nombre de rapports à gérer. Vous pouvez créer un rapport unique que les utilisateurs peuvent filtrer en fonction de leurs besoins, plutôt que de créer différents rapports pour chaque utilisateur. Par conséquent, un rapport associé à une application PowerPrompts peut afficher des données qui, sans PowerPrompts, nécessiteraient la création d'un ensemble de rapports.

Les usagers de rapports peuvent désormais utiliser un seul rapport pour de nombreuses interrogations ; ils n'ont pas besoin de comprendre les métadonnées associées au rapport lorsqu'ils sélectionnent les informations à afficher.

Documentation complémentaire

- « Définition de l'éditeur et du modèle HTML » (p. 11)
- « Lancement de PowerPrompts » (p. 11)
- « Création d'une application » (p. 12)
- « Mise en forme des pages finale, d'erreur et initiale » (p. 14)
- « Ajout d'une page » (p. 14)
- « Liaison de pages : aperçu » (p. 16)
- « Dynamos : aperçu » (p. 19)

Connaissances requises

PowerPrompts Developer Studio est un outil de développement d'applications. En tant que concepteur d'applications PowerPrompts, vous devez connaître :

- le langage HTML,
- le langage JavaScript,
- le langage SQL (si vous souhaitez utiliser des dynamos),
- les bases de données logiques (définitions de bases de données pour Impromptu),
- les serveurs Web.

Différences entre PowerPrompts 6.0 et la version Series 7

La version 6.0 et la version Series 7 de PowerPrompts présentent des différences considérables:

- Le langage de script désormais utilisé est ECMAScript (communément appelé JavaScript).
- Si vous utilisiez le code FormatUserVar, vous devez désormais utiliser le code suivant :

  ```
  <%"'"+ App.Variables("ListeÉtats").Join("','") + "'"%>
  ```

- Tous les index sont maintenant basés sur zéro. Dans la version 6.0, ils étaient basés sur un. Par exemple, l'index suivant provient de PowerPrompts 6.0 :

  ```
  <%GetDataCol(1)%>
  ```

  Dans PowerPrompts de la version Series 7, il a été remplacé par:

  ```
  <%=rs.Fields(0)%>
  ```

- Les appels de fonctions peuvent désormais être imbriqués.
- La boîte de dialogue *Gestionnaire de script* n'existe plus dans PowerPrompts de la version Series 7. Les conditions de scripts sont remplacées par des instructions IF dans la boîte de dialogue *Éditeur de script*, telles que :

  ```
  if ( GetUserVarAsString("ObjectifVentes") == "Y" )
  ```

- Les méthodes de rapports doivent être précédées de la méthode GetReport(), par exemple GetReport().AddDataItem.
- Le langage JavaScript fait la distinction entre les majuscules et les minuscules. Par exemple, App.RunDynamo est correct contrairement à aPP.rUNdYNAMO.

Mise à niveau des applications PowerPrompts 6.0 vers la version 7.1

PowerPrompts modifie vos applications PowerPrompts 6.0 pour qu'elles puissent être exécutées dans PowerPrompts 7.1. PowerPrompts ne déplace pas les fonctions JavaScript du côté client vers le côté serveur ou vice versa.

PowerPrompts 7.1 ne crée que des applications 7.1 mais peut lire et exécuter des applications 6.0 et 7.1.

Procédure de mise à niveau d'une application 6.0

1. Dans le menu *Démarrer*, cliquez sur *PowerPrompts*.
2. Dans le menu *Fichier*, cliquez sur *Ouvrir*.
3. Sélectionnez le fichier d'application (.xxm) dans le dossier d'application 6.0, puis cliquez sur *Ouvrir*.

   Votre application s'ouvre dans PowerPrompts Developer Studio.
4. Dans le menu *Fichier*, cliquez sur l'option *Enregistrer*.

   Un fichier de sauvegarde est créé lorsque l'application PowerPrompts 6.0 est ouverte dans PowerPrompts 7.1. Le nombre 60 est ajouté au nom du fichier de sauvegarde. Si vous ouvrez un fichier d'application PowerPrompts 6.0 appelé Exemple.xxm et que vous l'enregistrez, le fichier de sauvegarde s'appelle Exemple60.xxm.

L'application est enregistrée en tant qu'application PowerPrompts 7.1.

Applications PowerPrompts 6.0 déployées avec Impromptu Web Reports

Les applications PowerPrompts développées et publiées dans Impromptu Web Reports 6.0 fonctionnent sous Impromptu Web Reports 7.1 sans avoir besoin d'être modifiées ou à nouveau publiées. Impromptu Web Reports 7.1 ouvre l'application PowerPrompts 6.0 et la convertit en application PowerPrompts 7.1 à chaque nouvelle exécution. Pour gagner du temps en supprimant cette étape de conversion, ouvrez les applications dans PowerPrompts Developer Studio 7.1, enregistrez-les, puis publiez-les à nouveau dans Impromptu Web Reports 7.1.

Sécurité de PowerPrompts et intégration d'Access Manager

Introduction

Lors de l'exécution d'une application, PowerPrompts peut avoir besoin de communiquer avec des bases de données. Pour ce faire, PowerPrompts doit fournir la chaîne de connexion et les informations de sécurité appropriées. PowerPrompts vous permet de fournir ces informations de deux façons: l'incorporation d'informations de sécurité et l'intégration d'Access Manager.

Chaîne de connexion

La chaîne de connexion est constituée d'informations utilisées par le processus d'accès universel aux données (UDA) pour établir une connexion à une base de données. Cette chaîne contient des informations telles que le code de base de données, le type de la base de données (par exemple, Oracle, ODBC ou MS SQL). Elle permet également de spécifier si le code et le mot de passe utilisateur sont nécessaires. L'UDA peut ensuite déterminer la base de données à ouvrir et les informations qui doivent être transmises à cette dernière.

La chaîne de connexion d'une base de données n'est pas directement incorporée dans l'application PowerPrompts, ni dans les fichiers HTML de l'application. La base de données est identifiée par un nom logique et la chaîne de connexion peut être enregistrée dans le fichier Cognos.ini ou dans Access Manager.

Le fichier Cognos.ini est utilisé par les produits de Cognos antérieurs à Series 7. Les chaînes de connexion aux bases de données sont stockées dans ce fichier. Une base de données est répertoriée par son nom logique et une chaîne de connexion séparés par '='. Les codes d'utilisateur et les mots de passe ne sont pas enregistrés dans le fichier Cognos.ini. Ils doivent être fournis par l'application soit dans un script, soit via une demande.

Les chaînes de connexion sont enregistrées dans Access Manager en tant que propriété d'une source de données. Les administrateurs peuvent créer des objets de sources de données non liés à l'utilisateur, identifiés par un nom logique, comme pour les sources de données du fichier Cognos.ini. L'administrateur doit spécifier une chaîne de connexion pour chaque source de données et peut créer au moins un code d'accès. Ces codes d'accès sont ensuite communiqués à un ou plusieurs utilisateurs d'Access Manager.

Lorsque l'application PowerPrompts se connecte à une base de données, elle recherche la chaîne de connexion dans ces deux emplacements. Elle interroge tout d'abord Access Manager pour se connecter à la source de données. L'utilisateur doit pouvoir s'authentifier avec Access Manager et la source de données doit être répertoriée dans sa liste de référence. Si ces conditions ne sont pas remplies, PowerPrompts recherche ensuite la base de données logique dans le fichier Cognos.ini. Si la base de données y est répertoriée, PowerPrompts utilise cette chaîne de connexion. Si la source de données est introuvable dans les deux emplacements, une erreur s'affiche dans le navigateur.

Code d'utilisateur et mot de passe

Les codes d'utilisateur et les mots de passe ne sont pas traités de la même façon que les chaînes de connexion car ils contrôlent l'accès à la base de données. Ils ne peuvent pas être enregistrés sous forme non cryptée à un emplacement public, tel que dans le fichier Cognos.ini.

Le code d'utilisateur et le mot de passe de base de données peuvent être incorporés directement dans l'application, soit dans un fichier d'application (.xxm) si l'accès à la source de données est effectué via une dynamo, soit dans une page HTML si l'accès à la source de données est effectué via un objet Recordset.

Pour les dynamos, les codes d'utilisateur et les mots de passe sont spécifiés dans la page *Source de données de la dynamo* de l'Assistant de création de dynamos. Vous pouvez également modifier le code d'utilisateur et le mot de passe définis pour une dynamo. Pour ce faire, utilisez la boîte de dialogue *Source de données de la dynamo*. Cochez la case *Utiliser les informations d'authentification ci-dessous*, puis saisissez un code d'utilisateur et un mot de passe.

Pour spécifier un code d'utilisateur et un mot de passe pour un objet Recordset, les créateurs doivent définir un troisième et un quatrième paramètre pour la méthode Recordset.Open. Ces paramètres représentent respectivement le code d'utilisateur et le mot de passe.

Si un code d'utilisateur et un mot de passe sont spécifiés pour une dynamo ou un objet Recordset, PowerPrompts utilise uniquement ces valeurs et n'essaie pas d'effectuer l'authentification avec Access Manager. Si le code d'utilisateur et le mot de passe ne sont pas incorporés dans la dynamo ou dans l'objet Recordset, PowerPrompts essaie alors d'obtenir ces informations à partir d'Access Manager. L'utilisateur doit pouvoir s'authentifier et doit disposer des droits appropriés pour les informations de code d'accès à la source de données.

Le code d'utilisateur et le mot de passe ne sont pas toujours obligatoires (comme pour la base de données d'exemple Vacances et aventure). PowerPrompts essaie de se connecter à la base de données même si aucune information de sécurité n'est disponible.

Si un code d'utilisateur et un mot de passe sont requis mais qu'ils sont erronés ou non disponibles, une erreur s'affiche dans le navigateur.

Métadonnées

Des informations de sécurité et une chaîne de connexion sont nécessaires pour pouvoir extraire des métadonnées d'une base de données protégée. Le processus d'extraction de données de PowerPrompts est identique au processus d'extraction de données normales d'une base de données.

Paramètres du fichier d'initialisation

PowerPrompts contient un fichier d'initialisation (*xxxpowerprompts.ini)* dans lequel se trouvent les entrées suivantes.

ServerIdleLifeTime

Définit la durée maximale (en secondes) pendant laquelle un processus DataAccess peut rester inactif. Tout processus DataAccess restant inactif pendant une durée supérieure à cette limite est arrêté afin de libérer des ressources système. La valeur implicite est 900 secondes (15 minutes).

Port

Définit le numéro de port auquel le serveur DataAccess de PowerPrompts est connecté. La valeur implicite est 2425.

DataServersLimit

Définit le nombre maximal de processus DataAccess pouvant être exécutés simultanément pour le traitement de requêtes de données. Si cette valeur est 0, il n'existe aucune limite et autant de processus que nécessaire sont créés. Si cette valeur est un entier positif, le nombre de processus DataAccess n'excède jamais cette valeur plus 1. Par exemple, si vous définissez cette valeur à 5, six processus au maximum seront exécutés simultanément. La valeur implicite est 0.

Ajout d'un répertoire virtuel pour les images

Si vous souhaitez que des graphiques s'affichent sur les pages HTML de votre application PowerPrompts, vous pouvez définir un répertoire virtuel pour les images.

1. Créez le répertoire virtuel suivant sur votre serveur Web.

Alias	Répertoire	Accès
/images	Cognos\cer*n*\webcontent\images	Lecture

2. Placez vos images dans le dossier ..\webcontent\images.
3. Ajoutez les images aux pages HTML en utilisant l'attribut source de la balise d'image, par exemple :

```
<IMG SRC="/images/fichier1.gif">
```

4. Vérifiez que les images s'affichent dans votre application PowerPrompts.

Lancement de PowerPrompts

Description

PowerPrompts est disponible avec la version administrateur d'Impromptu.

Pour en savoir davantage sur l'installation d'Impromptu, reportez-vous au Guide d'installation.

Procédure

- Dans le menu *Démarrer*, cliquez sur *PowerPrompts*.

PowerPrompts Developer Studio s'ouvre et un espace de travail vide apparaît. Pour créer une application PowerPrompts, ajoutez des pages et définissez des liens entre celles-ci dans l'espace de travail PowerPrompts.

Documentation complémentaire

- « Création d'une application PowerPrompts » (p. 7)
- « Définition de l'éditeur et du modèle HTML » (p. 11)
- « Création d'une application » (p. 12)

Définition de l'éditeur et du modèle HTML

Description

Définissez l'éditeur HTML implicite que vous souhaitez utiliser. Par exemple, si vous connaissez bien le langage HTML, définissez le Bloc-Notes comme éditeur HTML implicite.

Utilisez les balises HTML classiques pour mettre en forme les pages PowerPrompts. Une page peut contenir tout type de mise en forme créé à l'aide de balises HTML. Vous pouvez mettre les pages en forme après les avoir ajoutées ou définir un modèle à appliquer à toutes les pages que vous ajoutez. Si vous définissez un modèle, toutes les pages que vous ajoutez à l'application (à l'exception de la page d'erreur et de la page finale) sont des copies du modèle. Vous pouvez ajouter des contrôles ou supprimer des éléments. Ceci vous permet de conserver une cohérence au niveau de la présentation de vos pages et de réduire le temps nécessaire à la mise en forme.

Remarques

- Le modèle HTML n'a aucune incidence sur les pages que vous importez dans une application.
- Vous pouvez aussi ajouter le contenu d'un fichier JavaScript (.js) à chaque page, à l'aide de l'instruction #include. Pour consulter un exemple, reportez-vous au manuel *À la découverte de PowerPrompts*.

Procédure

1. Dans le menu *Outils*, cliquez sur la commande *Options*.
2. Sélectionnez l'onglet *Répertoires*.
3. Dans la zone *Éditeur HTML*, spécifiez l'éditeur de texte HTML que vous souhaitez utiliser pour modifier les pages de votre application.
   Cet éditeur est utilisé à chaque fois que vous modifiez une page HTML depuis PowerPrompts Developer Studio.
4. Dans la zone *Fichier de modèle HTML*, spécifiez le modèle HTML à utiliser pour cette application, puis cliquez sur *OK*.
   Une fois le modèle HTML défini, toutes les pages ajoutées à l'application (à l'exception de la page d'erreur et de la page finale) sont des copies du modèle.

Documentation complémentaire

- « Création d'une application PowerPrompts » (p. 7)
- « Création d'une application » (p. 12)
- « Importation d'une page » (p. 14)
- « Utilisation d'un fichier JavaScript pour ajouter du code aux pages » (p. 12)

Utilisation d'un fichier JavaScript pour ajouter du code aux pages

Plutôt que de définir un modèle HTML, vous pouvez ajouter du code JavaScript et HTML d'un fichier JavaScript (.js) aux pages de l'application, à l'aide de l'instruction #include. Par exemple, vous pouvez ajouter un en-tête commun à toutes les pages ou le logo de l'entreprise contenu dans le fichier JavaScript.

Les exercices du didacticiel *À la découverte de PowerPrompts* utilisent le fichier Header.js pour appliquer la même présentation à toutes les pages de l'application.

Procédure

1. Dans le menu *Outils*, cliquez sur l'option *Éditeur HTML*.
   Votre éditeur HTML s'affiche.
2. Saisissez le code que vous souhaitez ajouter à toutes les pages de votre application.
   Par exemple, le code suivant ajoute le fichier Header.js :
   ```
   <%#include "Header.js"%>
   ```
3. Enregistrez le fichier. Assurez-vous qu'il porte l'extension .js.
4. Fermez l'éditeur HTML.
5. Placez le fichier .js dans le dossier de l'application PowerPrompts.

Pour en savoir davantage sur l'utilisation de l'instruction #include, reportez-vous à la section « Instruction #include et autres instructions du préprocesseur » (p. 42).

Documentation complémentaire

- « Définition de l'éditeur et du modèle HTML » (p. 11)

Création d'une application

Description

Organisez vos pages HTML dans l'espace de travail PowerPrompts Developer Studio.

Chaque application PowerPrompts créée comprend :

- une page initiale, qui est la première page que les usagers de rapports voient s'afficher.
- une page finale, qui initialise les scripts en fonction des sélections des usagers de rapports. Tous les chemins de navigation via l'application PowerPrompts doivent se terminer par la page finale et aucun lien de cette page ne peut désigner des éléments ne se trouvant pas dans la page finale.
- une page d'erreur, qui s'affiche si une erreur se produit pendant qu'une application PowerPrompts est en cours d'exécution. Aucun lien ne peut se trouver dans la page d'erreur ou désigner celle-ci.

Le chemin entre les pages initiale et finale peut contenir autant de pages que vous le souhaitez. Pour en savoir davantage sur la création de chemins de navigation, reportez-vous à la section « Liaison de pages : aperçu » (p. 16).

Procédure

1. Cliquez sur la commande *Nouveau* du menu *Fichier*.
   La boîte de dialogue *Nouvelle application* s'affiche.
2. Dans la zone *Nom de l'application*, saisissez le nom que vous souhaitez utiliser.
   La zone *Répertoire* spécifie automatiquement un sous-dossier du dossier d'application implicite. Le sous-dossier créé porte le même nom que votre application PowerPrompts ; il est créé lorsque vous enregistrez l'application.
   Le fichier d'application (.xxm) et les pages HTML (.htm) utilisés dans l'application sont stockés dans ce dossier. Les pages HTML ne doivent pas être supprimées du dossier d'application.
3. Dans la zone *Répertoire*, acceptez l'emplacement proposé ou spécifiez-en un nouveau, puis cliquez sur *OK*.
4. Pour tester votre application PowerPrompts et votre serveur Web et vous assurer de leur bon fonctionnement, cliquez sur *Exécuter*.
5. Si la page initiale s'affiche dans votre navigateur Web, fermez votre navigateur et retournez dans PowerPrompts Developer Studio.

Remarques

- Vous pouvez insérer des pages pour une application à n'importe quel endroit dans l'espace de travail. Vous pouvez également les déplacer à tout moment, les liens resteront intacts. Ces pages sont automatiquement créées en tant que fichiers HTML dans le dossier d'application.
- La boîte de dialogue *Nouvelle application* contient également la zone *Rapport à exécuter*. Vous n'avez pas besoin de sélectionner de rapport à exécuter avant d'avoir fini de créer votre application PowerPrompts. Pour en savoir davantage, reportez-vous à la section « Sélection d'un rapport pour tester l'application » (p. 24).

Documentation complémentaire

- « Ajout d'une page » (p. 14)
- « Création d'une application PowerPrompts » (p. 7)
- « Dynamos : aperçu » (p. 19)
- « Liaison de pages : aperçu » (p. 16)
- « Définition de l'éditeur et du modèle HTML » (p. 11)
- « Script : aperçu » (p. 23)

Mise en forme des pages finale, d'erreur et initiale

Si vous avez défini un modèle HTML, la page initiale est créée à partir de ce modèle. Si vous n'avez pas défini de modèle HTML, la page initiale contient uniquement le bouton *Suivant*, qui permet à l'usager de rapports d'atteindre la page suivante. Les pages finale et d'erreur ne sont pas créées à partir d'un modèle HTML. La page d'erreur exécute une méthode PowerPrompts qui affiche un message d'erreur à l'intention de l'usager du rapport. La page finale est créée à partir d'un formulaire HTML qui exécute un script pour le rapport. Pour consulter un exemple d'ajout de mise en forme aux pages, reportez-vous au manuel *À la découverte de PowerPrompts*.

Vous pouvez définir une nouvelle page initiale, mais vous ne pouvez pas modifier les pages finale et d'erreur, sauf leur nom.

Vous pouvez aussi ajouter le contenu d'un fichier JavaScript (.js) à chaque page, à l'aide de l'instruction #include. Pour consulter un exemple, reportez-vous au manuel *À la découverte de PowerPrompts*.

Documentation complémentaire

- « Création d'une application PowerPrompts » (p. 7)
- « Définition de l'éditeur et du modèle HTML » (p. 11)

Ajout d'une page

Description

Chaque page que vous ajoutez à l'application représente une page HTML distincte qui affichera une demande pour les usagers de rapports. Vous pouvez inclure autant de demandes que vous le souhaitez sur une page HTML.

Vous pouvez ajouter autant de pages HTML que vous le souhaitez dans une application PowerPrompts.

Procédure

1. Cliquez sur *Insérer une page*.
2. Cliquez sur l'espace de travail à l'endroit où vous souhaitez ajouter la page.
   La page est ajoutée et le nom est sélectionné.
3. Saisissez un nom unique pour la nouvelle page, puis appuyez sur [Entrée].

Documentation complémentaire

- « Création d'une application » (p. 12)
- « Suppression d'une page » (p. 15)
- « Modification d'une page » (p. 15)
- « Importation d'une page » (p. 14)
- « Liaison de pages : aperçu » (p. 16)

Importation d'une page

Description

Plutôt que d'ajouter une page blanche ou une copie du modèle HTML, vous pouvez importer une page HTML existante contenant les contrôles ou la mise en forme qui vous intéresse.

Conseil : Pour importer des pages à partir de l'Explorateur Windows, effectuez un copier-coller ou un glisser-déposer vers l'espace de travail.

Procédure

1. Dans le menu *Fichier*, cliquez sur *Importer un fichier HTML*.
2. Sélectionnez le fichier HTML à importer, puis cliquez sur le bouton *Ouvrir*.
   La page s'affiche dans l'espace de travail PowerPrompts et une copie du fichier HTML apparaît dans le dossier d'application.

Documentation complémentaire

- « Ajout d'une page » (p. 14)
- « Création d'une application » (p. 12)
- « Suppression d'une page » (p. 15)
- « Modification d'une page » (p. 15)
- « Liaison de pages : aperçu » (p. 16)

Modification d'une page

Description

Utilisez un éditeur HTML pour ajouter des contrôles, des tables, des couleurs et des arrière-plans aux pages HTML.

Pour consulter un exemple d'ajout de contrôles HTML à une page, reportez-vous au manuel *À la découverte de PowerPrompts*.

Procédure

1. Dans l'espace de travail, cliquez dans la page à modifier.
2. Dans le menu *Outils*, cliquez sur l'option *Éditeur HTML*.
   Votre page s'ouvre dans l'éditeur HTML implicite.
   Remarque : Vous pouvez également cliquer avec le bouton droit de la souris sur une page, puis cliquer sur *Éditer HTML* pour modifier la page.

Pour en savoir davantage sur le langage HTML, consultez la documentation de l'éditeur HTML.

Documentation complémentaire

- « Ajout d'une page » (p. 14)
- « Création d'une application » (p. 12)
- « Suppression d'une page » (p. 15)
- « Importation d'une page » (p. 14)
- « Liaison de pages : aperçu » (p. 16)
- « Définition de l'éditeur et du modèle HTML » (p. 11)

Suppression d'une page

Description

Supprimez les pages de l'application PowerPrompts dont vous n'avez plus besoin. Lorsque vous supprimez une page, vous supprimez également tous les liens qu'elle contient ou qui la désignent.

Procédure

1. Dans l'espace de travail, cliquez sur une page.

2. Cliquez sur le bouton *Supprimer*.
   PowerPrompts supprime la page et tous les liens qu'elle contient ou qui la désignent.

Conseil

- Pour supprimer plusieurs pages en même temps, sélectionnez-les et cliquez sur le bouton *Supprimer*.

Documentation complémentaire

- « Ajout d'une page » (p. 14)
- « Création d'une application » (p. 12)
- « Modification d'une page » (p. 15)
- « Importation d'une page » (p. 14)
- « Liaison de pages : aperçu » (p. 16)

Liaison de pages : aperçu

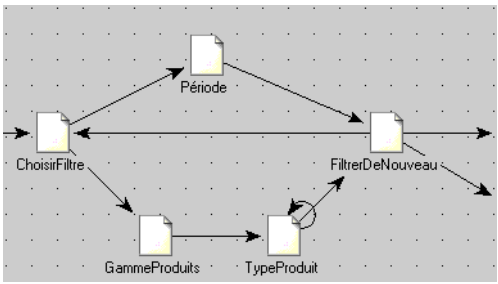
Les liens que vous ajoutez représentent les chemins de navigation possibles qui servent de guide aux usagers de rapports lors de la visualisation d'un rapport associé à une application PowerPrompts.

Vous pouvez avoir autant de liens que de pages dans l'application à condition que chaque page puisse atteindre la page finale. Vous pouvez créer un chemin de navigation qui :

- connecte les pages initiale et finale directement, sans branches ni boucles,
- affiche la même page plusieurs fois (boucle),
- propose des branches à partir d'une même page,
- contient une page liée à elle-même.

Un exemple de chemin de navigation en boucle serait un chemin allant de *ChoisirFiltre* vers *Période*, puis vers *AutreFiltre* et enfin de nouveau vers *ChoisirFiltre*.

Un exemple de chemin de navigation en branche serait un chemin allant de *ChoisirFiltre* vers *Période* ou *TypeProduit*.



Boîte de dialogue *Propriétés de la page*

La boîte de dialogue *Propriétés de la page* vous permet de modifier et de supprimer des liens. Lorsque vous ajoutez un lien, les deux pages sont connectées dans l'espace de travail. Si vous ouvrez la boîte de dialogue *Propriétés de la page* pour la page à partir de laquelle le lien commence, les propriétés de ce lien s'affichent. Tous les liens commençant sur cette page sont répertoriés dans l'onglet *Liens*. Chaque lien dispose d'une entrée dans les colonnes *Condition* et *Page*.

Lors de l'exécution, la liste des liens pour chaque page est évaluée du haut vers le bas. Le premier lien dont la condition est satisfaite correspond à la page suivante à ouvrir. Les liens situés en dessous de celui-ci ne sont pas évalués. Si la valeur <IMPLICITE> apparaît dans la colonne *Condition*, ce lien constitue le lien implicite. La condition du lien implicite est toujours satisfaite. Assurez-vous que le lien implicite est le dernier répertorié dans l'onglet *Liens*. Si vous placez des liens en dessous du lien implicite, les conditions de ces liens ne seront jamais vérifiées. Pour en savoir davantage sur les conditions des liens, reportez-vous à la section « Ajout d'une condition de lien » (p. 18).

Lorsque vous ajoutez un lien, la colonne *Condition* contient la valeur <IMPLICITE> puisque vous n'avez encore ajouté aucune condition de lien. Si vous supprimez une condition de lien, la valeur <IMPLICITE> s'affiche à nouveau dans la colonne *Condition* pour ce lien.

Conseil

- Pour déplacer le lien implicite (<IMPLICITE>), utilisez la flèche *Vers le bas*.

Documentation complémentaire

- « Ajout d'un lien par glissement » (p. 17)
- « Ajout d'un lien à l'aide de la boîte de dialogue » (p. 17)
- « Ajout d'une condition de lien » (p. 18)
- « Création d'une application » (p. 12)
- « Création d'une application PowerPrompts » (p. 7)
- « Suppression d'un lien » (p. 18)
- « Script : aperçu » (p. 23)

Ajout d'un lien par glissement

Description

Ajoutez un lien en le faisant glisser d'une page vers une autre dans l'espace de travail. Cette méthode permet de créer rapidement des applications PowerPrompts.

Vous pouvez également ajouter un lien dans l'onglet *Liens* de la boîte de dialogue *Propriétés de la page*.

Procédure

1. Cliquez sur l'option *Créer un lien*.
2. Faites glisser le lien d'une page vers une autre.

Documentation complémentaire

- « Ajout d'un lien à l'aide de la boîte de dialogue » (p. 17)
- « Ajout d'une condition de lien » (p. 18)
- « Suppression d'un lien » (p. 18)
- « Liaison de pages : aperçu » (p. 16)

Ajout d'un lien à l'aide de la boîte de dialogue

Description

Si une page contient plusieurs liens, créez les liens dans la boîte de dialogue *Propriétés de la page* puisque vous devez ajouter des conditions de liens afin que les usagers de rapports puissent sélectionner la page qui les intéresse. Une fois qu'une condition de lien a été ajoutée, sélectionnez la page à atteindre lorsque cette condition est remplie.

Vous pouvez également ajouter un lien par glissement entre deux pages.

Procédure

1. Cliquez sur l'option *Sélectionner une page*.
2. Cliquez avec le bouton droit de la souris dans la page à partir de laquelle vous souhaitez que le lien commence, puis cliquez sur *Propriétés*.
3. Cliquez sur l'onglet *Liens*.
4. Cliquez sur le bouton *Nouveau*.
   Une nouvelle ligne de lien apparaît.
5. Si vous souhaitez que ce lien contienne une condition, ajoutez une condition de lien.
6. Dans la colonne *Page*, sélectionnez la page à lier, puis cliquez sur *OK*.

Documentation complémentaire

- « Ajout d'un lien par glissement » (p. 17)
- « Ajout d'une condition de lien » (p. 18)
- « Suppression d'un lien » (p. 18)
- « Liaison de pages : aperçu » (p. 16)

Ajout d'une condition de lien

Description

Si une page comporte plusieurs liens, vous devez ajouter les conditions de liens afin que les usagers de rapports puissent sélectionner une page. Une fois les conditions de liens définies dans la boîte de dialogue *Propriétés de la page*, ajoutez le code HTML à la page afin que les usagers de rapports puissent spécifier la condition qui correspond à la page qu'ils veulent afficher. Lors de l'exécution, la liste des liens pour chaque page est évaluée du haut vers le bas. Le premier lien dont la condition est satisfaite correspond à la page suivante à ouvrir. Les conditions sont basées sur les variables de formulaires définies lors de la création de l'application. Par exemple, la condition TypeFiltre = "SelonPériode" du didacticiel *À la découverte de PowerPrompts* est évaluée comme valeur *Vrai* si vous sélectionnez le bouton *Filtrage selon la période* de la page *ChoisirFiltre*. Les liens situés en dessous de celui-ci ne sont pas évalués. Par conséquent, vous devez vous assurer que les liens sont dans le bon ordre.

Procédure

1. Cliquez sur l'option *Sélectionner une page*.
2. Cliquez avec le bouton droit de la souris dans la page pour laquelle vous souhaitez ajouter une condition de lien, puis cliquez sur *Propriétés*.
3. Cliquez sur l'onglet *Liens*.
4. Cliquez sur la colonne *Condition*, puis sur le bouton *Modifier*.
   La boîte de dialogue *Condition du lien* s'affiche.
5. Cliquez sur le bouton *Nouveau*.
   PowerPrompts recherche les valeurs de noms (par exemple, NAME="FilterType") dans la page sélectionnée et les affiche dans la zone *Nom*.
6. Dans la zone *Nom*, saisissez le nom de l'élément à évaluer dans l'expression. Par exemple, dans l'expression Prix < 10, le nom est Prix.
7. Dans la zone *Opérateur*, cliquez sur l'opérateur relationnel à utiliser dans l'expression. Par exemple, dans l'expression Prix < 10, il s'agit de l'opérateur inférieur à (<).
8. Dans la zone *Valeur*, saisissez la valeur à utiliser dans l'expression. Par exemple, dans l'expression Prix < 10, la valeur est 10.
9. Cliquez sur *OK* à deux reprises.

Conseil

- Si vous souhaitez utiliser plusieurs expressions dans votre condition de lien, ajoutez des expressions, puis liez-les avec les opérateurs logiques *ET* ou *OU*.

Documentation complémentaire

- « Ajout d'un lien par glissement » (p. 17)
- « Ajout d'un lien à l'aide de la boîte de dialogue » (p. 17)
- « Suppression d'un lien » (p. 18)
- « Liaison de pages : aperçu » (p. 16)

Suppression d'un lien

Description

Lors de la création de votre application PowerPrompts, vous pouvez repérer des liens dont vous n'avez plus besoin. Supprimez les liens dans l'onglet *Liens* de la boîte de dialogue *Propriétés de la page*. Si vous n'avez plus besoin de la page, supprimez-la ; tous les liens qu'elle contient ou qui la désignent sont également supprimés.

Procédure

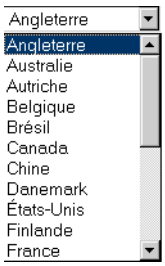
1. Cliquez sur l'option *Sélectionner une page*.
2. Cliquez avec le bouton droit de la souris dans la page à partir de laquelle le lien à supprimer commence, puis cliquez sur *Propriétés*.
3. Cliquez sur l'onglet *Liens*.
4. Cliquez sur le lien à supprimer.
   La page à laquelle le lien renvoie est répertoriée dans la colonne *Page*.
5. Cliquez sur *Supprimer*, puis sur *OK*.
   Le lien est supprimé de l'espace de travail.

Documentation complémentaire

- « Ajout d'un lien par glissement » (p. 17)
- « Ajout d'un lien à l'aide de la boîte de dialogue » (p. 17)
- « Ajout d'une condition de lien » (p. 18)
- « Suppression d'une page » (p. 15)
- « Liaison de pages : aperçu » (p. 16)

Dynamos : aperçu

Une dynamo permet de mettre en forme le résultat d'une interrogation SQL sur une base de données logique tel que défini dans le Gestionnaire de définition de bases de données. Par exemple, vous pouvez créer une dynamo pour extraire les pays que renferme une source de données. Une dynamo peut contenir plusieurs pays, selon le contenu de la source de données et les mises à jour. La liste apparaît lorsque les usagers de rapports affichent la page HTML qui contient la méthode App.RunDynamo.



Pour une dynamo, vous devez spécifier les éléments suivants :

- Source de données : la base de données logique à interroger.
- SQL : code SQL qui définit l'interrogation.
- Définition : code JavaScript qui met en forme les résultats.

Pour pouvoir interroger une base de données, vous devez disposer d'une base de données logique préalablement définie dans votre fichier Cognos.ini ou dans Access Manager. Si vous souhaitez utiliser des connexions ODBC, créez une base de données logique pour la connexion ODBC ou utilisez ADO directement avec la méthode JavaScript Server.CreateObject. Les anciennes applications PowerPrompts 6.0 qui utilisent des connexions ODBC fonctionnent correctement dans PowerPrompts, Series 7.

Pour en savoir davantage sur la création de bases de données logique, reportez-vous à l'*Aide du Gestionnaire de définition de bases de données*.

Boîte de dialogue *Gestionnaire de dynamos*

La boîte de dialogue *Gestionnaire de dynamos* permet d'ajouter, de modifier et de supprimer des dynamos. Toutes les dynamos créées disposent d'une entrée dans les colonnes *Nom* et *Source de données* et parfois dans la colonne *Description*. Vous pouvez copier les dynamos de cette boîte de dialogue, puis les coller dans vos pages HTML.

Remarque : Vous recevrez peut-être un message d'erreur lorsque vous tenterez d'ajouter une source de données dans un dynamo dans Windows 2000 ou Windows 2000 Professional. Dans ce cas, faites une recherche sur votre lecteur local et trouver le fichier MSSTDFMT.DLL. Assurez-vous que le fichier se trouve dans le répertoire \system32 sur votre ordinateur. Si vous ne parvenez pas à trouver ce fichier, communiquez avec votre administrateur système.

Conseil : Plutôt que de créer une dynamo, vous pouvez écrire du code JavaScript qui renvoie les valeurs d'une interrogation SQL sur les pages HTML. Pour consulter un exemple d'utilisation de code JavaScript au lieu d'une dynamo, reportez-vous au manuel *À la découverte de PowerPrompts*.

Documentation complémentaire

- « Création d'une dynamo à l'aide de l'assistant » (p. 20)
- « Création manuelle d'une dynamo » (p. 21)
- « Création d'une application » (p. 12)
- « Création d'une application PowerPrompts » (p. 7)

Création d'une dynamo à l'aide de l'assistant

Description

L'Assistant de création de dynamos vous guide lors de la création d'une dynamo qui utilise une source de données. Il est plus facile de créer une dynamo avec l'assistant que manuellement.

Procédure

1. Cliquez sur *Gestionnaire de dynamos*.
2. Cliquez sur *Assistant de création de dynamos*, puis suivez les instructions jusqu'à ce vous atteignez la page *Sélectionnez la source de données de la dynamo*.
3. Si vous souhaitez spécifier un code d'utilisateur et un mot de passe, cochez la case *Utiliser les informations d'authentification ci-dessous*, puis saisissez le code d'utilisateur et le mot de passe.
4. Dans la zone *Base de données logique*, sélectionnez la base de données logique à utiliser, puis cliquez sur *Suivant*.
5. Si vous souhaitez que seules les valeurs uniques soient extraites, cochez la case *Utiliser seulement des valeurs uniques*.
6. Sélectionnez la table contenant les données pour la dynamo, puis cliquez sur *Suivant*.

7. Si vous créez une liste, une case d'option ou une case à cocher, spécifiez le nom, la valeur et le libellé du contrôle HTML, puis cliquez sur *Suivant*.

Contrôle	Objectif
Nom du contrôle HTML	Nom utilisé dans les conditions. Dans l'exemple ci-dessous, le nom du contrôle HTML est TypeFiltre. <INPUT type="radio" NAME="Filter-Type" VALUE="SelonPériode">>
Valeur du contrôle HTML	Valeur utilisée dans les conditions. Dans l'exemple suivant, la valeur du contrôle HTML est SelonPériode. <INPUT type="radio" NAME="Filter-Type" VALUE="SelonPériode">
Libellé du contrôle	Donnée réellement affichée par la dynamo. Elle peut être identique à la valeur mais cela n'est pas nécessairement le cas.

   Si vous créez une table, dans la colonne *Titre HTML*, sélectionnez les colonnes à afficher dans la table, utilisez les boutons *Vers le haut* et *Vers le bas* pour établir l'ordre d'affichage, puis cliquez sur *Suivant*.
8. Dans la zone *Récapitulatif*, vérifiez les informations, puis cliquez sur *Terminer*.
9. Cliquez avec le bouton droit de la souris sur la dynamo, puis cliquez sur *Copier*.
10. Collez la dynamo dans la section <form> de votre page HTML, puis enregistrez la page.
11. Fermez la boîte de dialogue *Gestionnaire de dynamos*.

Documentation complémentaire

- « Création manuelle d'une dynamo » (p. 21)
- « Dynamos : aperçu » (p. 19)

Création manuelle d'une dynamo

Description

Vous pouvez créer des dynamos manuellement. Vous pouvez écrire du code SQL plus complexe lorsque vous créez une dynamo manuellement que lorsque vous la créez avec l'Assistant de création de dynamos.

Conseil : Vous pouvez également créer des dynamos qui n'utilisent pas de base de données logique. Par exemple, si votre application contient de nombreuses pages, vous pouvez créer une dynamo pour insérer un titre sur chaque page. Si vous souhaitez changer le titre, modifiez uniquement la définition de la dynamo, toutes les pages qui utilisent ce titre seront modifiées. Pour ce faire, créez une dynamo qui n'utilise pas de source de données, puis insérez le titre et le format HTML dans la boîte de dialogue *Définition de la dynamo*. Dans chaque page sur laquelle le titre doit apparaître, ajoutez la balise `<%=App.RunDynamo("nom dynamo")%>`.

Procédure pour spécifier la source de données de la dynamo

1. Cliquez sur *Gestionnaire de dynamos*.
2. Cliquez sur *Nouveau*.
   Une nouvelle ligne de dynamo apparaît.
3. Dans la colonne *Nom*, saisissez le nom de la nouvelle dynamo.

4. Cliquez sur *Modifier*, puis cliquez sur *Source de données*.
   La boîte de dialogue *Source de données de la dynamo* s'affiche.
5. Cliquez sur *N'utiliser aucune connexion à la base de données* si votre dynamo ne doit extraire aucune donnée.
6. Si vous souhaitez que les données soient extraites de manière dynamique, sélectionnez l'option *Utiliser la base de données logique ci-dessous*, puis sélectionnez le nom de la base de données logique dans la liste déroulante.
7. Si vous souhaitez spécifier un code d'utilisateur et un mot de passe, cochez la case *Utiliser les informations d'authentification ci-dessous*, puis saisissez le code d'utilisateur et le mot de passe.
8. Si vous souhaitez limiter le nombre d'enregistrements extraits, cochez la case *Inclure au plus*, spécifiez un nombre dans la zone *Enregistrements*, puis cliquez sur *OK*.
   La boîte de dialogue *Gestionnaire de dynamos* s'affiche à nouveau.

Procédure pour spécifier le code SQL et la définition de la dynamo

1. Cliquez sur *Modifier*, puis sur *SQL*.
   La boîte de dialogue *Code SQL de la dynamo* s'affiche.
2. Saisissez le code SQL pour extraire les données de la source de données. Attribuez le code SQL à la propriété SQL d'un objet Dynamo comme suit :
   ```
   Dynamo.SQL = ...
   ```
   L'exemple suivant renvoie les valeurs CodeGammeProduits et GammeProduits et les trie par GammeProduits.
   ```
   Dynamo.SQL = 'Select CodeGammeProduits, GammeProduits from GammeProduits order by
   GammeProduits';
   ```
   Remarque : Vous pouvez faire glisser des dossiers et des colonnes depuis le volet *Métadonnées* vers le volet *SQL*, ce qui vous évite d'avoir à les saisir.
3. Cliquez sur *OK*.
   La boîte de dialogue *Gestionnaire de dynamos* s'affiche à nouveau.
4. Cliquez sur *Modifier*, puis sur *Définition*.
   La boîte de dialogue *Définition de la dynamo* s'affiche.
5. Saisissez le code JavaScript que vous voulez utiliser.
   L'exemple suivant utilise les valeurs CodeGammeProduits comme valeurs de résultats HTML affichées dans une liste déroulante, mais affiche les valeurs GammeProduits.
   ```
   var sOutput = '<p><select name="ProdLine">';

   while(!rs.EOF)
   {
          sOutput = sOutput + "<option value='" + rs.Fields("CodeGammeProduits") + "'>"
   + rs.Fields("GammeProduits") + "</option>";
          rs.MoveNext();
   }

   sOutput += '</select></p>';
   return sOutput;
   ```
   L'instruction « rs.EOF » recherche la fin de l'ensemble des enregistrements. L'instruction « while (!rs.EOF) » dans son ensemble indique que les instructions entre accolades seront exécutées jusqu'à ce que la fin de l'ensemble des enregistrements soit atteinte.
6. Cliquez sur *OK* pour fermer la boîte de dialogue *Définition de la dynamo*.
7. Cliquez avec le bouton droit de la souris sur la dynamo, puis cliquez sur *Copier*.
8. Fermez la boîte de dialogue *Gestionnaire de dynamos*.
9. Collez la dynamo dans la section <form> de votre page HTML, puis enregistrez la page.

Documentation complémentaire

- « Création d'une dynamo à l'aide de l'assistant » (p. 20)
- « Dynamos : aperçu » (p. 19)

Script : aperçu

Un script permet de modifier le rapport associé à l'application PowerPrompts, en fonction des sélections effectuées par les usagers de rapports. Pour PowerPrompts, version Series 7, le script est en code JavaScript. Dans PowerPrompts 6.0, le script était une combinaison de méthodes PowerPrompts et de méthodes de rapports Impromptu.

Boîte de dialogue *Éditeur de script*

La boîte de dialogue *Éditeur de script* permet de créer, de modifier et de supprimer des scripts dans votre application PowerPrompts.

Vous ne pouvez pas valider le script tant que vous n'avez pas exécuté l'application. Si le script comporte des erreurs, celles-ci apparaîtront uniquement dans le navigateur Web.

Documentation complémentaire

- « Ajout d'un script » (p. 23)
- « Création d'une application » (p. 12)
- « Création d'une application PowerPrompts » (p. 7)

Ajout d'un script

Description

Utilisez la boîte de dialogue *Éditeur de script* pour ajouter un script. Un script est un code JavaScript qui modifie le rapport associé à l'application PowerPrompts. Par exemple, si un usager de rapports sélectionne un pays, vous pouvez créer un filtre de rapport pour afficher uniquement les enregistrements pour ce pays. Pour consulter un exemple de script, reportez-vous au manuel *À la découverte de PowerPrompts*.

Procédure

1. Cliquez sur *Éditeur de script*.
2. Saisissez le script contenant les méthodes et les constantes dont vous avez besoin, puis cliquez sur *OK*.

   Par exemple, le script ci-après filtre le rapport en utilisant la valeur CodePaysVente fournie par l'usager du rapport.

   ```
   if (App.Variables("CodePays"))
   {
           GetQuery().AndFilterBy("[\\Pays\\CodePaysVente] = " +
   App.Variables("CodePays"));
   }
   ```

   Conseil : Pour insérer des méthodes associées aux rapports Impromptu et des constantes, cliquez avec le bouton droit de la souris sur la zone *Éditeur de script*, puis cliquez sur l'élément souhaité. Pour insérer des références aux tables et aux colonnes, ouvrez Impromptu, copiez ces références dans la boîte de dialogue *Dossiers*, puis copiez-les dans la boîte de dialogue *Éditeur de script*.
3. Fermez la boîte de dialogue *Éditeur de script*.

Lorsque l'usager du rapport effectue une sélection définissant cette valeur, le script est exécuté avant l'ouverture du rapport.

Documentation complémentaire

- « Script : aperçu » (p. 23)

Test et déploiement de l'application : aperçu

Sélectionnez un rapport Impromptu avec lequel tester votre application PowerPrompts. Vérifiez et testez l'application et corrigez les éventuelles erreurs de script. Une fois que l'application fonctionne correctement, vous pouvez la distribuer.

Documentation complémentaire

- « Correction des erreurs de script » (p. 25)
- « Notification de l'administrateur Impromptu Web Reports » (p. 27)
- « Sélection d'un rapport pour tester l'application » (p. 24)
- « Test de l'application » (p. 25)
- « Vérification de l'application » (p. 24)
- « Affichage du script généré » (p. 26)

Sélection d'un rapport pour tester l'application

Description

Vous pouvez exécuter l'application PowerPrompts avec plusieurs rapports à condition que les scripts créés s'y appliquent. Pour pouvoir tester les scripts dans une application, vous devez spécifier un rapport.

Procédure

1. Dans le menu *Fichier*, cliquez sur l'option *Propriétés*.
   L'onglet *Général* de la boîte de dialogue *Propriétés de l'application* s'affiche.
2. Cliquez sur *Parcourir*, sélectionnez le rapport, puis cliquez sur *Ouvrir*.
   Le rapport sélectionné s'affiche dans la zone *Rapport à exécuter*.
3. Cliquez sur *OK*.

Documentation complémentaire

- « Correction des erreurs de script » (p. 25)
- « Notification de l'administrateur Impromptu Web Reports » (p. 27)
- « Test de l'application » (p. 25)
- « Vérification de l'application » (p. 24)

Vérification de l'application

Description

Chaque fois que vous enregistrez une application, PowerPrompts Developer Studio vérifie que l'application est correcte et affiche des messages d'avertissement et d'erreur, le cas échéant. Cependant, vous pouvez également utiliser la commande *Vérification de l'application* pour vérifier le chemin et les liens dans l'application sans enregistrer cette dernière.

Remarque : Pour ouvrir la rubrique d'aide associée à une erreur de l'application, cliquez avec le bouton droit de la souris sur l'erreur, puis cliquez sur *Afficher l'aide*.

Procédure

1. Ouvrez l'application à vérifier.
2. Dans le menu *Fichier*, cliquez sur l'option *Vérifier*.
   La boîte de dialogue *Vérification de l'application* s'affiche et répertorie, le cas échéant, les erreurs de navigation, les pages HTML manquantes et les liens redondants.
   Remarque : Vous pouvez laisser la boîte de dialogue *Vérification de l'application* ouverte pendant que vous corrigez les erreurs répertoriées.
3. Corrigez les erreurs et effectuez une nouvelle vérification.

Une fois toutes les erreurs corrigées, un message s'affiche indiquant que vous pouvez déployer l'application.

Documentation complémentaire

- « Correction des erreurs de script » (p. 25)
- « Notification de l'administrateur Impromptu Web Reports » (p. 27)
- « Sélection d'un rapport pour tester l'application » (p. 24)
- « Test de l'application » (p. 25)
- « Affichage du script généré » (p. 26)

Test de l'application

Description

Une fois l'application PowerPrompts terminée, sans erreur, vous pouvez la tester. L'application et le rapport qui en résulte apparaissent dans votre navigateur Web comme pour un usager de rapports.

Pour tester une application, les logiciels suivants doivent être installés sur votre ordinateur :

- un serveur Web,
- un navigateur Web,
- une application de visualisation de fichiers PDF, telle qu'Adobe Acrobat Reader.

Vous devez également spécifier le catalogue et les informations de connexion. Vous pouvez être amené à saisir votre classe d'utilisateurs, votre mot de passe de classe d'utilisateurs, votre code d'utilisateur de base de données et votre mot de passe de base de données, en fonction de la source de sécurité utilisée.

Problèmes de connexion de PowerPrompts à un catalogue

La connexion est impossible si les informations de connexion au catalogue fournies sont erronées, incomplètes ou qu'elles n'ont pas été spécifiées.

Procédure

1. Lancez votre serveur Web.
2. Cliquez sur *Exécuter*.
   La page initiale de l'application s'affiche dans votre navigateur Web.
3. Suivez les instructions que vous avez créées dans l'application et sélectionnez les informations que vous souhaitez voir apparaître dans le rapport.
4. Sur la dernière page, cliquez sur *Exécuter le rapport*.
   Si aucune erreur de script ne survient et que PowerPrompts se connecte au catalogue, le rapport s'ouvre au format .pdf dans votre navigateur Web.

Documentation complémentaire

- « Correction des erreurs de script » (p. 25)
- « Notification de l'administrateur Impromptu Web Reports » (p. 27)
- « Sélection d'un rapport pour tester l'application » (p. 24)
- « Vérification de l'application » (p. 24)
- « Affichage du script généré » (p. 26)

Correction des erreurs de script

Description

Au cours du test de l'application, des erreurs de script peuvent apparaître. Elles s'affichent dans le navigateur Web lorsque l'application est exécutée avec un rapport.

Conseil : Vérifiez que les guillemets contenus dans les scripts sont corrects. Vérifiez également que les noms du dossier du catalogue et des colonnes sont correctement orthographiés. Pour en savoir davantage, reportez-vous à la section « Expressions Impromptu dans PowerPrompts » (p. 102).

Procédure

1. Exécutez l'application PowerPrompts.
2. Dans le navigateur Web, lisez les descriptions des erreurs.
3. Corrigez les erreurs et exécutez à nouveau l'application.

Documentation complémentaire

- « Sélection d'un rapport pour tester l'application » (p. 24)
- « Notification de l'administrateur Impromptu Web Reports » (p. 27)
- « Test de l'application » (p. 25)
- « Vérification de l'application » (p. 24)
- « Affichage du script généré » (p. 26)

Affichage du script généré

Description

Vous pouvez également générer un script à appliquer au rapport en fonction des sélections effectuées au cours de l'exécution de l'application. Ceci permet de tester l'application PowerPrompts et de déboguer votre script.

Procédure

1. Dans le menu *Fichier*, cliquez sur l'option *Propriétés*.
2. Désélectionnez la case *Rapport à exécuter*, puis cliquez sur *OK*.
   Ce rapport n'est plus associé à l'application PowerPrompts.
3. Dans le menu *Fichier*, cliquez sur l'option *Enregistrer*.
   La boîte de dialogue *Vérification de l'application* s'affiche et indique qu'aucun rapport n'est défini pour être exécuté avec votre application.
4. Cliquez sur *Enregistrer*.
5. Cliquez sur *Exécuter*.
6. Sélectionnez les informations que vous souhaitez afficher dans le rapport.
7. Dans la page finale, cliquez sur *Exécuter le rapport*.

Un script, contenant les valeurs sélectionnées, s'affiche dans votre navigateur Web. Si les scripts contiennent des commentaires, ces derniers s'affichent également.

Conseils

- Pour que la boîte de dialogue *Vérification de l'application* ne s'affiche pas, dans le menu *Outils*, cliquez sur *Options*, puis désélectionnez la case *Ne pas tenir compte des avertissements pendant la vérification*.
- Pour ne pas être invité à enregistrer les modifications avant de tester l'application Power-Prompts, dans le menu *Outils*, cliquez sur *Options*, puis sur *Enregistrer sans demande de confirmation*.

Documentation complémentaire

- « Correction des erreurs de script » (p. 25)
- « Notification de l'administrateur Impromptu Web Reports » (p. 27)
- « Sélection d'un rapport pour tester l'application » (p. 24)
- « Test de l'application » (p. 25)
- « Vérification de l'application » (p. 24)

Notification de l'administrateur Impromptu Web Reports

Une fois l'application PowerPrompts créée et testée, informez-en l'administrateur Impromptu Web Reports. Lorsqu'un usager de rapports ouvre un rapport associé à une application PowerPrompts, cette dernière s'exécute dans son navigateur Web. L'usager est alors invité à spécifier les informations qui doivent apparaître dans le rapport.

Pour en savoir davantage sur les informations dont doit disposer l'administrateur des rapports une fois que l'application PowerPrompts est créée, reportez-vous à l'*Aide sur le portail Web de Cognos* d'Upfront.

Documentation complémentaire

- « Correction des erreurs de script » (p. 25)
- « Sélection d'un rapport pour tester l'application » (p. 24)
- « Test de l'application » (p. 25)
- « Vérification de l'application » (p. 24)
- « Affichage du script généré » (p. 26)

Chapitre 2 : Procédures

Filtrage d'un rapport pour plusieurs valeurs

Description

Il est possible d'appliquer un filtre à un rapport pour plusieurs valeurs. Par exemple, vous pouvez souhaiter afficher les chiffres de ventes à la fois pour le Canada et les États-Unis.

Procédure

1. Ajoutez le contrôle HTML <select> avec l'attribut multiple à votre page ou dans une dynamo, par exemple :

   ```
   <select name="Countries" multiple>...</select>
   ```

2. Écrivez le script pour ajouter le filtre à l'aide de la méthode GetQuery().AndFilterBy(). Pour les données n'étant pas des chaînes, le script est simple, par exemple :

   ```
   GetQuery().AndFilterBy("[CodePays] in (" + App.Variables("CodePays") + ")");
   ```

   Remarque : Dans le script, utilisez "in" au lieu de "=" car "in" prend en charge plusieurs valeurs.

   Si la donnée est une chaîne, faites-la apparaître entre guillemets simples. Pour ce faire, utilisez la propriété Join de l'objet StringList, par exemple :

   ```
   GetQuery().AndFilterBy("[Ville] in ('" +  App.Variables("Villes").Join("','") +
   "')");
   ```

Exemple

Cet exemple permet d'ajouter une liste déroulante des pays dans la page, puis de filtrer le rapport en fonction des pays sélectionnés.

Page HTML

```
<select name="CountryCd" multiple>
<%
var rs = new Recordset();
rs.Open("Select CodePaysVente,Pays from Pays", "VAV");
while(!rs.EOF)
{
    Response.Write("<option value='" + rs.Fields("CodePaysVente") + "'>" +
rs.Fields("Pays") + "</option>");
    Response.Write("<br>");
    rs.MoveNext();
}
rs.Close();
%>
</select>
```

Script

```
if (App.Variables("Pays"))
{
    GetQuery().AndFilterBy("[\\Pays\\Codepaysvente] in (" + App.Variables("CodePays") +
")");
}
```

Documentation complémentaire

- « Ajout d'un filtre de classe d'utilisateurs d'Impromptu à des listes de valeurs » (p. 31)
- « Création d'une liste en cascade sur une seule page » (p. 33)
- « Création d'un récapitulatif des choix des utilisateurs » (p. 30)
- « Création d'une liste de valeurs saisie au clavier » (p. 32)

Enregistrement des choix des utilisateurs entre les sessions

Description

Les utilisateurs peuvent enregistrer les choix qu'ils effectuent, puis les utiliser à nouveau lorsqu'ils exécutent l'application une nouvelle fois.

Pour enregistrer les choix des utilisateurs, écrivez-les dans une base de données en utilisant l'objet Connection de PowerPrompts ou ADO si votre ordinateur fonctionne sous Windows.

Procédure

1. Écrivez les valeurs App.Variables dans la base de données dans la partie supérieure de chaque page HTML de votre application. Une base de données simple est constituée d'un tableau à 3 colonnes.
2. Chaque fois qu'une page est générée, exécutez une interrogation de sélection pour déterminer si la base de données contient des valeurs pour les demandes de cette page. Si tel est le cas, initialisez les contrôles avec les valeurs correctes.

Exemple

Cet exemple permet de créer une base de données simple où sont stockées trois valeurs : 1) nom d'utilisateur, 2) nom de variable et 3) valeur.

```
<%
var conn = new Connection();
for (var sName in App.Variables)
{
    conn.Execute("INSERT INTO ÉtatUtilisateur (NomUtilisateur, NomVariable, Valeur)
VALUES ( '"+ User.UserName + "','" + sName + "','" + App.Variables(sName) + "')");
}
%>
```

Documentation complémentaire

- « Création de différentes présentations pour différents thèmes Upfront » (p. 35)
- « Création d'un récapitulatif des choix des utilisateurs » (p. 30)

Création d'un récapitulatif des choix des utilisateurs

Description

Vous pouvez créer un récapitulatif des choix déjà effectués par l'utilisateur.

Procédure

- Dans la page finale de l'application PowerPrompts, ajoutez un code JavaScript qui affiche les variables définies par l'utilisateur. Le code suivant affiche, dans une table HTML, le pays qu'un utilisateur a choisi d'afficher :

```
<br><b>Table de variables sélectionnées</b>
<table>
<table border=2 bgcolor=#80FFFF>
<tr width="15%">
<th>Nom de variable</th>
<th>Valeur</th>
</tr>
<%
if (App.Variables("CodePays"))
{
    var rs = new Recordset;
    rs.Open("SELECT Pays FROM Pays WHERE CodePaysVente=" +
App.Variables("CodePays"),"VAV");
    if (!rs.EOF)
    {
        Response.Write("<tr><td>Pays</td><td>" + rs.Fields("Pays") + "</td></tr>");
    }
}
...
```

Remarque : Le chapitre 2 du manuel *À la découverte de PowerPrompts* comprend un exemple de code JavaScript pour afficher les choix des utilisateurs. Le code de cet exemple se trouve également dans le fichier *pwp_html.txt* situé dans le dossier *emplacement_installation*\Exemples\PowerPrompts.

Documentation complémentaire

- « Création de différentes présentations pour différents thèmes Upfront » (p. 35)
- « Enregistrement des choix des utilisateurs entre les sessions » (p. 30)

Ajout d'un filtre de classe d'utilisateurs d'Impromptu à des listes de valeurs

Description

Vos catalogues peuvent déjà contenir des filtres de classe d'utilisateurs. Il est possible d'utiliser ces informations dans vos applications PowerPrompts. Par exemple, si un catalogue contient une classe d'utilisateurs par pays, vous pouvez filtrer les données de chaque rapport en fonction de la classe à laquelle appartient l'utilisateur en cours. Par conséquent, lors de la création d'une application PowerPrompts pour un rapport de ce catalogue, vous pouvez souhaiter voir s'afficher une liste déroulante des villes qui ne comprenne que les villes correspondant à la classe d'utilisateurs de l'utilisateur. Pour ce faire, vous pouvez utiliser un objet Query.

Procédure

- Utilisez l'objet Query pour envoyer une instruction SQL à l'objet Recordset avec le filtre approprié en fonction de la classe d'utilisateurs en cours, tel que dans l'exemple ci-dessous.

Exemple

Cet exemple permet de créer une liste déroulante des villes en fonction du filtre de classe d'utilisateurs.

```
<%
var myQuery = new Query();
myQuery.AddColumn("[\\Pays\\Ville]","Ville");
var sql = myQuery.SQL;
var rs = new Recordset();
rs.Open(sql,"VENTESVA");
Response.Write("<select name=city>");
while(!rs.EOF)
{
/   Response.Write("<option>" + rs.Fields(0) + "</option>");
    rs.MoveNext();
}
Response.Write("</select>");
%>
```

Documentation complémentaire

- « Création d'une liste en cascade sur une seule page » (p. 33)
- « Création d'une liste de valeurs saisie au clavier » (p. 32)
- « Filtrage d'un rapport pour plusieurs valeurs » (p. 29)

Création d'une liste de valeurs saisie au clavier

Description

Il est possible de créer une liste de valeurs saisie au clavier en ajoutant un contrôle des entrées de texte à une page HTML. Une fois la valeur extraite du contrôle des entrées par PowerPrompts, elle peut être utilisée soit dans le script final soit dans une expression Impromptu.

Procédure

1. Il est possible de créer une liste de valeurs saisie au clavier en ajoutant un contrôle des entrées de texte à une page HTML.
2. Une fois la valeur extraite du contrôle des entrées, elle peut être utilisée soit dans le script final soit dans une expression Impromptu.

Exemple

Cet exemple permet de filtrer le rapport final en fonction du code de commande saisi par l'usager.

Page HTML

```
<html>
<body>
<h1>Entrée du code de commande</h1>
Veuillez entrer un code de commande.<p>
<form method="post">
Order Code<br>
<input type="text" name="order">
<input type="submit" name="Enter">
</form>
</body>
</html>
```

Script

```
GetQuery().AndFilterBy("[Code de commande] = " + App.Variables("commande"));
```

Documentation complémentaire

- « Ajout d'un filtre de classe d'utilisateurs d'Impromptu à des listes de valeurs » (p. 31)
- « Création d'une liste en cascade sur une seule page » (p. 33)
- « Filtrage d'un rapport pour plusieurs valeurs » (p. 29)

Création d'une liste en cascade sur une seule page

Description

Il est possible de filtrer une liste de valeurs en fonction d'une valeur sélectionnée dans une autre liste de valeurs, sur la même page. Par exemple, vous pouvez filtrer une liste de villes en sélectionnant tout d'abord un pays.

Procédure

1. Écrivez le code JavaScript côté client pour créer deux listes de valeurs.
2. Ajoutez deux liens depuis la page de listes de valeurs.
3. Spécifiez le premier lien en tant que condition, btn=Next, et pour qu'il renvoie à la page suivante de l'application.
4. Spécifiez que le lien implicite renvoie à sa propre page. Il s'agit d'un lien de renvoi automatique.

   Dans l'application, lorsque vous sélectionnez un pays, la page est envoyée et la liste de villes appropriée est générée. Cliquez sur le bouton *Suivant* pour faire apparaître la page suivante.

Exemple

Cet exemple permet de filtrer une liste de villes en fonction du pays sélectionné dans la liste correspondante.

Page HTML

```
<html>
<body>
<h1>Liste en cascade</h1>
<form method="post">

Pays<br>
<select name="country" OnChange="document.forms[0].submit()">
<%
// Recherchez la liste des pays
var rs = new Recordset();
rs.Open("SELECT pays FROM pays ORDER BY pays","VAV");
// Entrez chaque pays dans la liste
var sSelectedText;
var sPreviousCountry = App.Variable("pays");
while(!rs.EOF)
{
    // Ce code présélectionne le pays choisi au préalable
    if (sPreviousCountry == rs.Fields(0))
    {
        sSelectedText = " selected ";
    }
    else
    {
        sSelectedText = "";
    }

    Response.Writeln("<option " + sSelectedText + " >" + rs.Fields(0));
    rs.MoveNext();
}
%>
</select>
<p>
Ville<br>
<select name="city">
<%
```

```
// Recherchez la liste des villes filtrée par pays
var rs2 = new Recordset();
rs2.Open("SELECT ville FROM pays WHERE pays = '" + App.Variables("pays") + "' ORDER BY
ville","VAV");
while (!rs2.EOF)
{
    Response.Writeln("<option>" + rs2.Fields(0));
    rs2.MoveNext();
}
%>
</select>
<p>
<input type=submit name=btn value=Suivant>
</form>
</body>
</html>
```

Documentation complémentaire

- « Ajout d'un filtre de classe d'utilisateurs d'Impromptu à des listes de valeurs » (p. 31)
- « Création d'une liste de valeurs saisie au clavier » (p. 32)
- « Filtrage d'un rapport pour plusieurs valeurs » (p. 29)

Utilisation d'ADO dans PowerPrompts

Description

Vous pouvez utiliser les objets de données ActiveX (ADO) de Microsoft dans les applications PowerPrompts pour accéder à une source de données ODBC. Utilisez la méthode Server.CreateObject pour instancier un objet COM, puis en utiliser les propriétés et méthodes.

Remarque : Le langage JavaScript ne prend pas en charge le concept des propriétés et méthodes implicites. Par conséquent, vous devez explicitement appeler chaque méthode ADO.

Procédure

1. Écrivez le code pour créer un objet ADO.
2. Ajoutez ce code à l'une de vos pages HTML, enregistrez les modifications et exécutez l'application.

Exemple

Cet exemple permet de créer un objet de connexion ADO, de se connecter à une source de données ODBC, de renseigner un ensemble d'enregistrements, puis de renseigner la première zone de votre page HTML.

```
<%
var oCmd = Server.CreateObject("ADODB.Command");
oCmd.ActiveConnection = "DB_GREATOUTDOORS";
oCmd.CommandText = "SELECT * FROM Pays";
var oRS = oCmd.Execute();
while (!oRS.EOF)
{
    Response.Writeln(oRS.Fields("Pays").Value + "<br>");
    oRS.MoveNext();
}
%>
```

Extraction du code d'utilisateur et du mot de passe de base de données depuis Access Manager

Description

Vous n'avez pas besoin d'ajouter des informations de code d'utilisateur et de mot de passe de base de données dans votre application PowerPrompts. Ces informations peuvent être extraites depuis Access Manager. Pour ce faire, vous devez tout d'abord configurer correctement Access Manager. Assurez-vous que tous les utilisateurs qui se serviront de l'application PowerPrompts ont accès à la base de données requise et qu'ils disposent du code d'accès approprié pour chaque base de données. Veuillez vous reporter à la documentation d'Access Manager pour en savoir davantage sur cette opération.

Par la suite, ne fournissez pas de code d'utilisateur ou de mot de passe lorsque vous appelez la méthode Recordset.Open.

Procédure

1. Configurez Access Manager.
2. Appelez la méthode Recordset.Open sans fournir de code d'utilisateur ni de mot de passe.
3. Lors de la création de dynamos, désactivez la case *Utiliser les informations d'authentification ci-dessous* pour pouvoir extraire le code d'utilisateur et le mot de passe depuis Access Manager.

Exemple

Cet exemple demande à Access Manager s'il existe un code d'accès de base de données pour l'utilisateur en cours. S'il en existe un, il sera utilisé.

```
rs.Open("SELECT * FROM PAYS","Base de données");
```

Création de différentes présentations pour différents thèmes Upfront

Description

Upfront prend en charge différents thèmes. Chacun d'entre eux peut avoir une présentation différente. Les usagers peuvent sélectionner la présentation qui leur convient. Implicitement, les thèmes des applications PowerPrompts ont la même présentation. Toutefois, vous pouvez créer vos applications de façon à ce que leur présentation diffère en fonction du thème sélectionné par l'usager. Pour ce faire, accédez à la propriété Upfront.Theme et modifiez la page basée sur cette propriété.

Procédure

- Écrivez du code qui modifie vos pages HTML en fonction du thème Upfront sélectionné par l'utilisateur.

Exemple

Cet exemple permet de créer deux thèmes appelés « Corporate » et « Flashy ».

Page HTML

```
<html>
<head>
<%
    switch (Upfront.Theme)
    {
        case "Corporate":
%>
    <link rel="stylesheet" type="text/css" href="/corp.css">
<%
        break;

        case "Flashy":
%>
    <link rel="stylesheet" type="text/css" href="/flashy.css">
<%
        break;
    }
%>
</head>
<body>
<%
    switch (Upfront.Theme)
    {
        case "Corporate":
%>
    <h1>Thème Corporate</h1>
    Le thème Corporate est une présentation de type entreprise.
<%
        break;

        case "Flashy":
%>
    <h1>Thème Flashy</h1>
    Le thème Flashy est une présentation de type fantaisie.
<%
        break;
    }
%>

</body>
</html>
```

Création d'une seule application PowerPrompts pour plusieurs langues

Vous pouvez créer votre application PowerPrompts pour qu'elle apparaisse avec du texte dans la langue appropriée à l'usager.

1. Sur votre page HTML, au lieu d'entrer du texte, créez une variable en langage JavaScript pour chaque mot ou groupe de mots.
2. Utilisez la méthode Response.Write pour que le texte de la variable apparaisse à l'emplacement où vous souhaitez que le texte apparaisse dans la page.
3. En haut de la page, demandez à Upfront la langue actuellement sélectionnée et définissez les variables sur le texte approprié.

Exemple

Cet exemple retourne du texte en fonction de la langue sélectionnée par l'usager.

Page HTML

```
<html>
<head>
<%
```

```
// Sélectionnez la langue et enregistrez-la
var slang = Upfront.Language;
var sTitle;
var sCountry;
var SelectText;
if (sLang == "en")
{
    sTitle = "Welcome Page";
    sCountry = "Country";
    sSelectText = "Select a Country";
}
else if (slang == "fr")
{
    sTitle = "Page de Bienvenue";
    sCountry = "Pays";
    sSelectText = "Choisir un Pays";
}
else
{
    // Implicitement, l'anglais est utilisé
    sTitle = "Welcome Page";
    sCountry = "Country";
    sSelectText = "Select a Country";
}
%>
</head>
<body>
<h1><%=sTitle%></h1>
<%=sSelectText%><p>
<%=sCountry%>
<select name=Country>
</select>
</body>
</html>
```

Chapitre 3 : Utilisation du langage JavaScript dans PowerPrompts

Vous pouvez utiliser le langage JavaScript dans vos pages HTML côté serveur et côté client.

Le langage JavaScript côté client est exécuté au niveau du navigateur. Consultez votre documentation sur JavaScript côté client pour obtenir des informations supplémentaires sur les règles de syntaxe. Le langage JavaScript côté serveur est exécuté au niveau de PowerPrompts et est utilisé pour créer le contenu d'une page et accéder aux objets de PowerPrompts et d'Impromptu Web Reports.

Vous trouverez ci-après quelques règles qui vous permettront d'écrire du code JavaScript côté serveur.

JavaScript côté serveur

- Le langage JavaScript côté serveur respecte dans son intégralité la norme ECMA-262.
- Lorsque vous insérez du code JavaScript côté serveur, vous devez le faire précéder d'un signe supérieur à et le faire suivre d'un signe inférieur à (<%...%>) afin que le serveur Powerprompts puissent reconnaître qu'il s'agit de code JavaScript.
- Pour ajouter des commentaires à votre code JavaScript vous devez appliquer les conventions utilisées en langage C++ (double barre oblique //) ou en langage C (/*…*/).

  Par exemple,

```
<%
/* Pour exécuter une dynamo qui renvoie un nombre et l'affecte à une chaîne */
var sReturn = App.RunDynamo("MaDynamo2");
if (sReturn > 0)
{
    Response.Write("<b>Commandes retournées :" + sReturn +"</b>");
}
%>
```

- Le langage JavaScript fait la distinction entre les majuscules et les minuscules. Par exemple, Response.Write est correct alors que RESPONSE.write ne l'est pas.
- Vous pouvez insérer un signe égal (=) dans le code Javascript entre les balises de signe inférieur à et de signe supérieur à (<%...%>) comme raccourci pour Response.Write.

  Par exemple,

```
<%=Request.Variables%>
```

  est identique à

```
<%Response.Write(Request.Variables)%>
```

Tableau principal des objets JavaScript côté serveur

Utilisez les objets côté serveur répertoriés ci-dessous pour créer le contenu d'une page et interagir avec les composants PowerPrompts et Impromptu Web Reports. Pour en savoir davantage sur le langage JavaScript côté serveur, reportez-vous à la section « Utilisation du langage JavaScript dans PowerPrompts » (p. 39).

Objet	Propriétés	Méthodes
« Objet App » (p. 43)	« Propriété BackURL » (p. 44) « Propriété CurrentPage » (p. 45)	« Méthode RunDynamo » (p. 48)

Objet	Propriétés	Méthodes
	« Propriété Errors » (p. 45)	
	« Propriété FinalURL » (p. 46)	
	« Propriété IsTestMode » (p. 46)	
	« Propriété Path » (p. 47)	
	« Propriété ReportScript » (p. 47)	
	« Propriété Variables » (p. 47)	
« Objet Connection » (p. 49)		« Méthode Execute » (p. 49)
« Objet Field » (p. 49)	« Propriété HTMLEncodedValue » (p . 50)	« Méthode Operator() (Zone) » (p. 54)
	« Propriété Index » (p. 50)	
	« Propriété Name » (p. 51)	
	« Propriété Type » (p. 52)	
	« Propriété Value » (p. 53)	
« Objet Query » (p. 54)	« Propriété SQL » (p. 56)	« Méthode AddColumn » (p. 56)
		« Méthode AndFilterBy » (p. 57)
		« Méthode AndSummaryFilterBy » (p. 57)
		« Méthode AssociateColumn » (p. 58)
		« Méthode GroupBy » (p. 58)
		« Méthode OrFilterBy » (p. 59)
		« Méthode OrSummaryFilterBy » (p. 59)
		« Méthode RemoveColumn » (p. 60)
		« Méthode SetDistinct » (p. 60)
		« Méthode SetPromptValue » (p. 61)

Objet	Propriétés	Méthodes
		« Méthode SortBy » (p. 61)
« Objet Recordset » (p. 62)	« Propriété CurrentRecordIndex » (p. 62)	« Méthode Close » (p. 65)
	« Propriété EOF » (p. 63)	« Méthode MoveNext » (p. 66)
	« Propriété Fields » (p. 64)	« Méthode Open » (p. 67)
	« Propriété MaxRecords » (p. 65)	
« Objet Request » (p. 68)	« Propriété Cookies » (p. 68)	
	« Propriété ServerVariables » (p. 69)	
	« Propriété Variables » (p. 47)	
« Objet Response » (p. 70)	« Propriété ContentType » (p. 71)	« Méthode AppendCookie » (p. 72)
		« Méthode AppendHeader » (p. 72)
		« Méthode Clear » (p. 72)
		« Méthode ClearContent » (p. 72)
		« Méthode ClearHeaders » (p. 72)
		« Méthode Redirect » (p. 73)
		« Méthode Write » (p. 73)
		« Méthode WriteFile » (p. 74)
		« Méthode WriteIn » (p. 74)
« Objet Server » (p. 75)	« Propriété ScriptTimeout » (p. 76)	« Méthode CreateObject » (p. 76)
		« Méthode FormatNumber » (p. 77)
		« Méthode HTMLEncode » (p. 77)
		« Méthode URLDecode » (p. 78)
		« Méthode URLEncode » (p. 78)
« Objet StringList » (p. 79)	« Propriété Count » (p. 79)	« Méthode Contains » (p. 80)
		« Méthode Item » (p. 81)
		« Méthode Join » (p. 82)

Objet	Propriétés	Méthodes
		« Méthode Operator() (Liste de chaînes) » (p. 82) « Méthode toString » (p. 83)
« Objet Upfront » (p. 83)	« Propriété Language » (p. 84) « Propriété Locale » (p. 84) « Propriété Theme » (p. 84)	« Méthode ExecuteCommand » (p. 85) « Méthode GetPageFragment » (p. 85)
« Objet User » (p. 85)	« Propriété Description » (p. 86) « Propriété Email » (p. 86) « Propriété Telephone » (p. 87) « Propriété UserClass » (p. 87) « Propriété UserClasses » (p. 88) « Propriété UserName » (p. 88)	

Instruction #include et autres instructions du préprocesseur

instruction #include

Utilisez l'instruction #include pour inclure d'autres codes JavaScript dans la page. Vous pouvez utiliser cette instruction pour partager du code entre les pages de votre application et également entre plusieurs applications. Le chemin d'accès spécifié se rapporte toujours au répertoire de l'application. Dans l'exemple 1, utilisez le chemin d'accès relatif lorsque vous souhaitez disposer d'un répertoire commun pour les fichiers JavaScript partagés sur un serveur.

Exemple 1

L'exemple ci-dessous illustre comment utiliser l'instruction #include lorsque le fichier SaveState.js se trouve dans le même dossier que l'application.

```
<%
#include "SaveState.js"
%>
```

Exemple 2

L'exemple ci-dessous illustre comment utiliser l'instruction #include lorsque le fichier SaveState.js se trouve dans le dossier CommonScripts, placé sous le dossier dans lequel se trouve l'application.

```
<%
#include "../CommonScripts/SaveState.js"
%>
```

Instructions du préprocesseur

Outre l'instruction #include, vous pouvez utiliser d'autres instructions du préprocesseur :

- #define
- #ifdef
- #else
- #endif

L'instruction #define est utilisée pour déclarer les variables communes, telles que :

```
#define NOM_SOCIETE Cognos
```

Les trois autres types d'instructions sont utilisées pour créer des instructions de définition conditionnelles.

Exemple

L'exemple ci-dessous illustre l'utilisation d'une instruction de définition conditionnelle pour vérifier certaines conditions sous UNIX.

```
#ifdef _UNIX_
    Response.Write("N'est pas pris en charge sous UNIX");
#else
    var ocom = Server.CreateObject("ADODB.Connection");
#endif
```

Constantes prédéfinies

Il existe deux constantes prédéfinies que vous pouvez utiliser pour tester des applications :

- TEST_MODE
- _UNIX_

Utilisez la constante prédéfinie TEST_MODE si vous souhaitez tester l'application PowerPrompts à l'aide du client Impromptu. Utilisez la constante prédéfinie _UNIX_ lorsqu'un serveur UNIX tente d'ouvrir une application PowerPrompts.

Objet App

Renvoie des informations sur l'utilisateur actuel dans l'application PowerPrompts.

Il s'agit d'un objet qui ne peut pas être créé.

Propriété	Description
Propriété BackURL	Renvoie l'adresse URL dans Upfront. Cette propriété vous permet de revenir dans Upfront à partir de votre application PowerPrompts. Utilisez cette propriété lorsque vous effectuez une action sur un formulaire.
Propriété CurrentPage	Renvoie le nom de la page courante.
Propriété Errors	Renvoie une liste de messages d'erreur pour la requête en cours sur le serveur.
Propriété FinalURL	Renvoie l'adresse URL dans UpFront après avoir quitter une application PowerPrompts. Utilisez cette méthode dans la page finale lorsque vous effectuez une action sur un formulaire.
Propriété IsTestMode	Renvoie une valeur indiquant si l'application PowerPrompts est exécutée en mode test ou non. Renvoie la valeur *Vrai* si l'application est exécutée à partir de PowerPrompts Developer Studio et renvoie la valeur *Faux* pour tous les autres cas. Permet d'effectuer la mise au point de votre application.
Propriété Path	Renvoie le chemin d'accès au dossier dans lequel se trouve le fichier d'application PowerPrompts (.xxm). Cette propriété s'avère très utile pour la méthode Response.WriteFile.
Propriété ReportScript	Renvoie le script qui sera envoyé vers Impromptu. Cette propriété est uniquement disponible en mode test et dans la page finale. Dans tous les autres cas, une chaîne vide sera renvoyée. Permet d'effectuer la mise au point de votre application.
Propriété Variables	Renvoie l'état de l'application. Il s'agit d'une collection de variables.

Méthode	Description
Méthode RunDynamo	Exécute la dynamo spécifiée pour générer une chaîne. Utilisez cette méthode dans vos pages HTML.

Propriété BackURL

Renvoie l'adresse URL dans Upfront. Cette propriété vous permet de revenir dans Upfront à partir de votre application PowerPrompts. Elle fonctionne uniquement pour des applications déployées qui sont exécutées à partir d'Impromptu Web Reports. Cette propriété ne fonctionne pas en mode test car Upfront n'est pas utilisé.

Utilisez cette propriété lorsque vous effectuez une action sur un formulaire.

Cette propriété est en lecture seule.

Type

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet de renvoyer le nom de l'utilisateur dans Upfront lorque celui-ci clique sur le bouton *Annuler*.

```
<form method="post" action="<%=App.BackURL%>">
<input type="submit" value="Annuler">
</form>
```

S'applique à :

Objet App

Propriété CurrentPage

Renvoie le nom de la page courante.

Cette propriété est en lecture seule.

Type

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet d'indiquer le nom de la page dans l'en-tête de premier niveau.

```
<!--Utiliser le nom de la page comme en-tête-->
<h1><%= App.CurrentPage %></h1>
```

Exemple de résultat

```
ChoixPays
```

S'applique à :

Objet App

Propriété Errors

Renvoie une liste de messages d'erreur pour la requête en cours sur le serveur.

Cette propriété est en lecture seule et fonctionne uniquement dans la page d'erreur.

Type

Liste de chaînes

Exemple

```
L'exemple de syntaxe ci-dessous permet d'indiquer toutes les erreurs, chacune étant
séparée par une ligne horizontale.
<%
// Fonctionne uniquement dans la page d'erreur.
for (var i = 0; i < App.Errors.Count; i++)
{
    if ( i > 0 )
    {
        Response.Writeln( "<hr>" );
    }
Response.Write(<p><pre>Server.HTMLEncode( App.Errors( i ) )</pre></p>)
}
%>
```

Exemple de résultat

```
Erreur à la ligne 10 de la page [ChoixPays]

<%=App.RunDynamo("Pays")%>
---------------------------------------------------------------
Échec de l'appel de la méthode [App.RunDynamo]

Error near no filename:10 [RunDynamo()].
      from no filename:10 [:Global Initialization:()]
---------------------------------------------------------------
Erreur à la ligne 5 de la dynamo [Pays] pendant l'exécution de la définition.
---------------------------------------------------------------
TypeError 1406: Variable is not a function type.
Error near no filename:5 [Global Code].
```

S'applique à :

Objet App

Propriété FinalURL

Renvoie l'adresse URL dans UpFront après avoir quitter une application PowerPrompts. Utilisez cette méthode dans la page finale lorsque vous effectuez une action sur un formulaire.

Cette propriété est en lecture seule.

Type

Chaîne

Exemple

L'exemple de syntaxe ci-dessous illustre comment créer un bouton permettant d'exécuter le rapport.

```
<form action="<%=App.FinalURL%>">
<input type = "submit" value = "Terminer">
</form>
```

S'applique à :

Objet App

Propriété IsTestMode

Renvoie une valeur indiquant si l'application PowerPrompts est exécutée en mode test ou non. Renvoie la valeur *Vrai* si l'application est exécutée à partir de PowerPrompts Developer Studio et renvoie la valeur *Faux* pour tous les autres cas. Permet d'effectuer la mise au point de votre application.

Cette propriété est en lecture seule.

Type

Booléen

Exemple

L'exemple ci-dessous illustre comment créer une zone de texte vide lorsque l'application est testée.

```
<!--Afficher uniquement le script du rapport si l'application est testée-->
<%
if (App.IsTestMode);
{
%>
<textarea>
<%=Server.HTMLEncode(App.ReportScript)%>
</textarea>
<%
}
%>
```

S'applique à :

Objet App

Propriété Path

Renvoie le chemin d'accès au dossier dans lequel se trouve le fichier d'application PowerPrompts (.xxm). Ce chemin d'accès comprend une barre oblique de séparation à la fin de chaque élément du chemin. En fonction du système d'exploitation sous lequel fonctionne PowerPrompts, il peut s'agir soit d'une barre oblique, soit d'une barre oblique inverse. Cette propriété s'avère utile pour la méthode Response.WriteFile.

Cette propriété est en lecture seule.

Type

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet d'indiquer le chemin d'accès à l'application.

```
<%Response.Write(App.Path);%>
```

Exemple de résultat

```
C:\Applications PowerPrompts\Représentants\
```

S'applique à :

Objet App

Propriété ReportScript

Renvoie le script qui sera envoyé vers Impromptu. Cette propriété est uniquement disponible en mode test et dans la page finale. Dans tous les autres cas, une chaîne vide sera renvoyée. Permet d'effectuer la mise au point de votre application.

Cette propriété est en lecture seule.

Type

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet de créer une zone de texte dans laquelle s'affiche le script.

```
<!--Afficher le script du rapport dans une zone de texte-->
<textarea>
<%=Server.HTMLEncode(App.ReportScript)%>
</textarea>
```

S'applique à :

Objet App

Propriété Variables

Renvoie l'état de l'application. Il s'agit d'une collection de variables. Contrairement aux variables de requêtes, les variables d'application sont toujours présentes entre chaque requête. Utilisez cette propriété pour extraire les variables définies dans l'application. Utilisez la propriété Request.Variables uniquement si vous avez besoin de paires de valeurs/noms de formulaire non traitées.

Cette propriété est en lecture seule.

Type

Liste de chaînes

Exemple

L'exemple de syntaxe ci-dessous permet de renvoyer le code pays du pays que vous avez sélectionné dans la page *ChoixPays*.

```
<%
// Extraire un contrôle de texte correspondant au CodePays si celui comporte une valeur,
sinon
// appliquer la valeur par défaut du texte
if (App.Variables("CodePays"))
{
    Response.Writeln("<input type='text' name='text1' value='" +
App.Variables("CodePays") + "'");
}
else
{
    Response.Writeln("<input type='text' name='text1' value='Valeur implicite'>");
}
%>
```

Exemple de résultat

```
16
```

S'applique à :

Objet App

Méthode RunDynamo

Exécute la dynamo spécifiée pour générer une chaîne. Utilisez cette méthode dans vos pages HTML. Vous ne pouvez pas utiliser cette méthode pour les boîtes de dialogue *Définition de la dynamo* ou *Code SQL de la dynamo*.

Paramètres

nomDynamo : Chaîne, obligatoire

Type de données renvoyées

Chaîne

Exemple 1

L'exemple de syntaxe ci-dessous permet d'exécuter MaDynamo1 et de définir sa présentation dans la page.

```
<!--Appliquer une dynamo créée à l'aide de l'assistant de création de modèles
dynamiques-->
<%=App.RunDynamo("MaDynamo1")%>
```

Exemple 2

L'exemple de syntaxe ci-dessous permet d'exécuter MaDynamo2 et si la valeur renvoyée est supérieure à zéro, un message s'affiche.

```
<%
var iReturn = parseInt(App.RunDynamo("MaDynamo2"));

if (iReturn > 0 )
{
    Response.Write("Commandes retournées : " + iReturn);
}
%>
```

Exemple de résultat

```
Commandes retournées : 4
```

S'applique à :

Objet App

Objet Connection

Objet qui modifie votre base de données grâce à l'utilisation d'instructions SQL. Cela vous permet de vous connecter à une source de données et d'exécuter des instructions SQL sans renvoyer de résultats.

Il s'agit d'un objet qui peut être créé.

Méthode	Description
Méthode Execute	Exécute l'instruction SQL qui modifie votre base de données.

Méthode Execute

Exécute l'instruction SQL qui modifie votre base de données.

Si l'exécution de cette méthode échoue, une page d'erreur s'affiche.

Paramètres

Le paramètre SQL correspond à l'instruction SQL à exécuter. Le paramètre nomBD correspond au nom de la base de données logique Cognos.

Il existe deux paramètres de chaînes facultatifs. Le paramètre codeUtilisateurBD correspond au code d'utilisateur de base de données à utiliser lors de la connexion à la base de donnée. Le paramètre motdepasseBD correspond au mot de passe de base de données pour ce code d'utilisateur.

SQL : Chaîne, obligatoire

nomBD : Chaîne, obligatoire

codeUtilisateurBD : Chaîne, facultatif

motdepasseBD : Chaîne, facultatif

Exemple

L'exemple de syntaxe ci-dessous permet d'ajouter les valeurs Jean Dupont et Programmeur à la base de données.

```
<%
var conn = new Connection();
conn.Execute("INSERT INTO Utilisateur(NomUtilisateur,DescriptionUtilisateur)
VALUES('Jean Dupont','Programmeur')", "Utilisateur");
Response.Write("<b>Interrogation réussie</b>");
%>
```

S'applique à :

Objet Connection

Objet Field

Un Objet Recordset se compose d'une collection Fields contenant les objets Field. Chaque Objet Field correspond à une colonne de donnée dans l'objet Recordset.

Il s'agit d'un objet qui ne peut pas être créé.

Propriété	Description
Propriété HTMLEncodedValue	Renvoie la valeur de la colonne dans les résultats pour la ligne courante, sauf si cette valeur est codée en HTML.
Propriété Index	Renvoie la position de la colonne dans les résultats. Il s'agit d'une valeur qui commence à partir de zéro.
Propriété Name	Renvoie le nom de la colonne dans les résultats.
Propriété Type	Renvoie le type de données de la zone.
Propriété Value	Renvoie la valeur de la colonne dans les résultats pour la ligne courante.

Méthode	Description
Méthode Operator() (Zone)	Indique l'opérateur d'index utilisé pour accéder aux éléments de la collection. Vous pouvez définir un index numérique ou une clé de chaîne.

Propriété HTMLEncodedValue

Renvoie la valeur de la colonne dans les résultats pour la ligne courante, sauf si cette valeur est codée en HTML. Le code Javascript équivalent est

```
Server.HTMLEncode(rs.Fields(0).Value)
```

Cette propriété est associée de manière implicite à l'objet Field. Les données de cette propriété sont de type Chaîne quel que soit le type de données d'origine de la base de données. Si vous souhaitez utiliser des valeurs numériques, utilisez les méthodes standard JavaScript pour convertir les chaînes de caractères en valeurs numériques. Si la valeur dans la base de données est de type *NUL*, alors aucune valeur ne sera renvoyée par JavaScript.

Cette propriété est en lecture seule.

Type

Chaîne

Exemple 1

```
<%=rs.Fields(0).HTMLEncodedValue%>
```

Exemple 2

```
<%=rs.Fields(0)%>
```

S'applique à :

Objet Field

Propriété Index

Renvoie la position de la colonne dans les résultats. Il s'agit d'une valeur qui doit commencer à 0, c'est-à-dire que l'index de la première colonne sera l'index 0, l'index de la seconde colonne sera l'index 1, et ainsi de suite. Par exemple, l'instruction SQL "SELECT Pays, CodePaysVente FROM Pays" comporte deux zones. Par exemple, à la zone *Pays* correspond l'index 0 et à la zone *CodePaysVentes* correspond l'index 1.

Cette propriété est en lecture seule.

Type

Nombre

Exemple

```
<%
// Extraire la valeur de la colonne, ainsi que son nom et l'index correspondant
var rs = new Recordset();
rs.Open("Select Pays,CodePaysVente from Pays","VAV");
// Extraire uniquement un tableau lorsque des résultats sont renvoyés
if (!rs.EOF)
{
    Response.Writeln("<table><tr>");
    // Afficher les en-têtes de tableau
    for (var i = 0; i < rs.Fields.Count; i++)
    {
        Response.Write("<td>" + rs.Fields(i).Name + " Index : " + rs.Fields(i).Index +
"</td>");
    }
    Response.Writeln("</tr>");
    while (!rs.EOF)
    {
        Response.Write("<tr>");
        for (var j = 0; j < rs.Fields.Count; j++)
        {
            Response.Write("<td>" + rs.Fields(j).Value + "</td>");
            rs.MoveNext();
        }
        Response.Writeln("</tr>");
    }
    Response.Writeln("</table>");
}
else
{
    Response.Writeln("Aucune donnée.<p>");
}
%>
```

Exemple de résultat

```
Index Pays : 0 Index CodePaysVente : 1
France 2
États-Unis 4
Autriche 6
Pays-Bas 8
Angleterre 10
Japon 12
Corée 14
Australie 17
Danemark 19
Mexique 21
Finlande 23
```

S'applique à :

Objet Field

Propriété Name

Renvoie le nom de la colonne dans les résultats. Par exemple, l'instruction SQL "SELECT Pays, Code_Pays AS Code FROM Pays" comporte deux zones. La première zone s'appelle *Pays* et la seconde s'appelle *Code*.

Cette propriété est en lecture seule.

Type

Chaîne

Exemple

```
<%
// Extraire la valeur de la colonne, ainsi que son nom et l'index correspondant
var rs = new Recordset();
rs.Open("Select Pays,CodePaysVente from Pays","VAV");
// Extraire uniquement un tableau lorsque des résultats sont renvoyés
if (!rs.EOF)
{
    Response.Writeln("<table><tr>");
    // Afficher les en-têtes de tableau
    for (var i = 0; i < rs.Fields.Count; i++)
    {
        Response.Write("<td>" + rs.Fields(i).Name + " Index : " + rs.Fields(i).Index +
"</td>");
    }
    Response.Writeln("</tr>");
    while (!rs.EOF)
    {
        Response.Write("<tr>");
        for (var j = 0; j < rs.Fields.Count; j++)
        {
            Response.Write("<td>" + rs.Fields(j).Value + "</td>");
            rs.MoveNext();
        }
        Response.Writeln("</tr>");
    }
    Response.Writeln("</table>");
}
else
{
    Response.Writeln("Aucune donnée.<p>");
}
%>
```

Exemple de résultat

```
Index Pays : 0 Index CodePaysVente : 1
France 2
États-Unis 4
Autriche 6
Pays-Bas 8
Angleterre 10
Japon 12
Corée 14
Australie 17
Danemark 19
Mexique 21
Finlande 23
```

S'applique à :

Objet Field

Propriété Type

Renvoie le type de données de la zone. Les constantes numériques suivantes peuvent être utilisées :

- ftDate
- ftDateTime
- ftInterval
- ftNumeric
- ftString
- ftTime
- ftUnknown

Cette propriété est en lecture seule.

Type

Nombre

Exemple

```
<%
// Extraire la valeur de la colonne, ainsi que son nom et l'index correspondant
var rs = new Recordset();
rs.Open("Select * from Pays","VAV");
// Extraire les noms des colonnes et les types de données correspondants
for (var i = 0; i < rs.Fields.Count; i++)
{
var sText = rs.Fields(i).Name + ": ";
    switch(rs.Fields(i).Type)
    {
        case ftDate:
            sText += "Date";
            break;
        case ftDateTime:
            sText += "Date/Heure";
            break;
        case ftInterval:
            sText += "Intervalle";
            break;
        case ftNumeric:
            sText += "Numerique";
            break;
        case ftString:
            sText += "Chaîne";
            break;
        case ftTime:
            sText += "Heure";
            break;
        case ftUnknown:
            sText += "Inconnu";
            break;
    }
Response.Writeln(sText + "<br>");
}
%>
```

Exemple de résultat

```
CodePaysVente : Nombre
Pays : Chaîne
CodeISO3Lettres : Chaîne
CodeISO2Lettres : Chaîne
CodeISO3Chiffres : Chaîne
NomMonnaie : Chaîne
EnEurosDepuis : Date/Heure
```

S'applique à :

Objet Field

Propriété Value

Renvoie la valeur de la colonne dans les résultats pour la ligne courante. Si la valeur dans la base de données est de type *NUL*, la valeur de type *nul* est renvoyée par JavaScript.

Les données de cette propriété sont de type Chaîne quel que soit le type de données d'origine de la base de données. Si vous souhaitez utiliser des valeurs numériques, utilisez les méthodes standard JavaScript pour convertir les chaînes de caractères en valeurs numériques.

Cette propriété est en lecture seule.

Type

Chaîne

Exemple

```
<%
// Afficher la valeur de la zone ou la chaîne "NUL" si la zone ne contient pas de valeur.
var rs = new Recordset();
rs.Open("Select CodePaysVente from Pays","VAV");
var sValue = rs.Fields( 0 ).Value;
Response.Write( ( sValue == null ) ? "NUL" : sValue );
%>
```

Exemple de résultat

```
1
```

S'applique à :

Objet Field

Méthode Operator() (Zone)

Indique l'opérateur d'index utilisé pour accéder aux éléments de la collection. Vous pouvez définir un index numérique ou une clé de chaîne.

Paramètres

Index : Numérique, obligatoire

ou

Clé : Chaîne, obligatoire

Type de données renvoyées

Élément de la collection

Exemple

```
<%
// Extraire la valeur de la colonne, ainsi que son nom et l'index correspondant
var rs = new Recordset();
rs.Open("Select * from Pays","VAV");
// Indiquer les deux manières d'utiliser la méthode Operator()
Response.Write(rs.Fields(0) + " est la valeur de la première colonne dans les
enregistrements.");
Response.Write("<br>");
Response.Write(rs.Fields("Pays") + " est la valeur de la colonne Pays dans les
enregistrements.");
%>
```

Exemple de résultat

```
1 est la valeur de la première colonne dans les enregistrements.
France est la valeur de la colonne Pays dans les enregistrements.
```

S'applique à :

Objet Field

Objet Query

Permet d'extraire des instructions SQL pour un rapport Impromptu. L'instruction SQL contiendra tous les filtres des classes d'utilisateurs et les qualificatifs de base de données définis dans le catalogue. Vous pouvez utiliser l'instruction SQL dans une dynamo ou une liste de valeurs pour les contrôles des utilisateurs. Toutes les tables pour lesquelles l'accès est refusé et que vous spécifiez dans le catalogue Impromptu ne sont pas prises en compte.

Vous pouvez créer l'objet Query en utilisant le chemin d'accès complet d'un rapport Impromptu. Cela signifie que l'objet Query renvoie l'interrogation principale pour ce rapport. ProwerPrompts ne peut pas accéder aux interrogations des sous-rapports. Si vous souhaitez commencer avec une interrogation vierge, n'insérez pas d'objet dans l'instruction du constructeur pour l'interrogation, par exemple,

```
var myQuery = new Query();
```

Il s'agit d'un objet qui peut être créé.

Propriété	Description
Propriété SQL	Renvoie l'instruction SQL pour l'objet Query, y compris les filtres du catalogue Impromptu, en fonction de la classe d'utilisateurs de l'utilisateur actuel.

Méthode	Description
Méthode AddColumn	Ajoute une colonne du catalogue Impromptu ou un calcul à l'interrogation.
Méthode AndFilterBy	Ajoute une expression de filtre de détail à l'interrogation. S'il existe déjà un filtre de détail, cette expression est ajoutée au filtre à l'aide de l'opérateur AND (ET).
Méthode AndSummaryFilterBy	Ajoute une expression de filtre de récapitulatif à l'interrogation. S'il existe déjà un filtre de récapitulatif, cette expression est ajoutée au filtre à l'aide de l'opérateur AND (ET).
Méthode AssociateColumn	Ajoute une association à une colonne dans l'interrogation.
Méthode GroupBy	Ajoute un regroupement à l'interrogation pour une colonne spécifiée.
Méthode OrFilterBy	Ajoute une expression de filtre de détail à l'interrogation. S'il existe déjà un filtre de détail, cette expression est ajoutée au filtre à l'aide de l'opérateur OR (OU).
Méthode OrSummaryFilterBy	Ajoute une expression de filtre de récapitulatif à l'interrogation. S'il existe déjà un filtre de récapitulatif, cette expression est ajoutée au filtre à l'aide de l'opérateur OR (OU).
Méthode RemoveColumn	Supprime une colonne dans l'interrogation et le rapport.
Méthode SetDistinct	Indique que l'interrogation contient uniquement des éléments distincts.

Méthode	Description
Méthode SetPromptValue	Définit une valeur pour une demande nommée dans l'interrogation.
Méthode SortBy	Outre les tris déjà définis dans l'interrogation, trie l'interrogation en fonction de la colonne spécifiée.

Propriété SQL

Renvoie l'instruction SQL pour l'objet Query, y compris les filtres du catalogue Impromptu, en fonction de la classe d'utilisateurs de l'utilisateur actuel.

L'instruction SQL renvoyée commence toujours par un point d'exclamation (!). Ce point d'exclamation indique à l'objet Recordset que l'instruction SQL a été créée à l'aide de l'objet Query.

Cette propriété est en lecture seule.

Type

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet d'ouvrir le rapport pwp_didacticiels_js.imr, qui se trouve dans le même dossier que l'application PowerPrompts. Renvoie ensuite l'instruction SQL du rapport dans l'objet Recordset. Tous les filtres des classes d'utilisateurs sont appliqués avant que l'instruction SQL soit renvoyée.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_didacticiels_js.imr");
var rs = new Recordset();
rs.Open(myQuery.SQL, "VAV");
%>
```

S'applique à :

Objet Query

Méthode AddColumn

Ajoute une colonne du catalogue Impromptu ou un calcul à l'interrogation. Pour en savoir davantage sur la création d'expressions pour le catalogue Impromptu, reportez-vous à la section « Expressions Impromptu dans PowerPrompts » (p. 102).

Le paramètre Nom correspond au nom de la colonne (ou du calcul) et doit être unique dans l'interrogation. Le paramètre Expression correspond à l'expression du catalogue Impromptu qui définit la colonne.

Paramètres

Nom : Chaîne, obligatoire

Expression : Chaîne, obligatoire

Type de données renvoyées

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet d'ajouter une colonne au rapport avant de renvoyer l'instruction SQL.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_didacticiels_js.imr");
myQuery.AddColumn("Année","[année([Commandes\\DateCommande])]");
rs.Open(myQuery.SQL, "VAV");
%>
```

S'applique à :

Objet Query

Méthode AndFilterBy

Ajoute une expression de filtre de détail à l'interrogation. S'il existe déjà un filtre de détail, cette expression est ajoutée au filtre à l'aide de l'opérateur AND (ET). Pour en savoir davantage sur la création d'expressions pour le catalogue Impromptu, reportez-vous à la section « Expressions Impromptu dans PowerPrompts » (p. 102).

Le paramètre filtreDétail correspond à l'expression de filtre de détail à ajouter.

Paramètres

filtreDétail : Chaîne, obligatoire

Type de données renvoyées

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet de filtrer le rapport par année avant de renvoyer l'instruction SQL.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_didacticiels_js.imr");
myQuery.AndFilterBy("année([Commandes\\DateCommande]) in (1999,2000)");
rs.Open(myQuery.SQL, "VAV");
%>
```

S'applique à :

Objet Query

Méthode AndSummaryFilterBy

Ajoute une expression de filtre de récapitulatif à l'interrogation. S'il existe déjà un filtre de récapitulatif, cette expression est ajoutée au filtre à l'aide de l'opérateur AND (ET). Pour en savoir davantage sur la création d'expressions pour le catalogue Impromptu, reportez-vous à la section « Expressions Impromptu dans PowerPrompts » (p. 102).

Le paramètre filtreRécapitulatif correspond à l'expression de filtre de récapitulatif à ajouter.

Paramètres

filtreRécapitulatif : Chaîne, obligatoire

Type de données renvoyées

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet de filtrer le rapport par année avant de renvoyer l'instruction SQL.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_didacticiels_js.imr");
myQuery.AndSummaryFilterBy("année([Commandes\\DateCommande]) in (1999,2000)");
rs.Open(myQuery.SQL, "VAV");
%>
```

S'applique à :

Objet Query

Méthode AssociateColumn

Ajoute une association à une colonne dans l'interrogation.

Le paramètre colonneNommée correspond au nom de la colonne à associer. Le paramètre colonneGroupée correspond au nom de la colonne groupée qui doit être associée à la première colonne.

Paramètres

colonneNommée : Chaîne, obligatoire

colonneGroupée : Chaîne, obligatoire

Type de données renvoyées

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet d'associer la colonne *Ventes totales* avec la colonne *Pays* avant de renvoyer l'instruction SQL.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_didacticiels_js.imr");
myQuery.AssociateColumn("Ventes totales","Pays");
rs.Open(myQuery.SQL, "VAV");
%>
```

S'applique à :

Objet Query

Méthode GroupBy

Ajoute un regroupement à l'interrogation pour une colonne spécifiée.

Le paramètre colonneGroupée correspond au nom de la colonne. Le paramètre ordreTri correspond soit à l'ordre croissant soit à l'ordre décroissant qui définit l'ordre dans lequel la colonne doit être triée.

Cette méthode associe un groupe à tout groupe présent dans le rapport.

Paramètres

colonneGroupée : Chaîne, obligatoire

ordreTri : Chaîne (croissant ou décroissant), obligatoire

Type de données renvoyées

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet de grouper la colonne *NomFournisseur* avant de renvoyer l'instruction SQL.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_didacticiels_js.imr");
myQuery.GroupBy("NomFournisseur","décroissant");
rs.Open(myQuery.SQL, "VAV");
%>
```

S'applique à :

Objet Query

Méthode OrFilterBy

Ajoute une expression de filtre de détail à l'interrogation. S'il existe déjà un filtre de détail, cette expression est ajoutée au filtre à l'aide de l'opérateur OR (OU). Pour en savoir davantage sur la création d'expressions pour le catalogue Impromptu, reportez-vous à la section « Expressions Impromptu dans PowerPrompts » (p. 102).

Le paramètre filtreDétail correspond à l'expression de filtre de détail à ajouter.

Paramètres

filtreDétail : Chaîne, obligatoire

Type de données renvoyées

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet d'ajouter un filtre de détail avant de renvoyer une instruction SQL.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_didacticiels_js.imr");
myQuery.OrFilterBy("[\\Produits\\Produit] commencepar 'VA'");
rs.Open(myQuery.SQL, "VAV");
%>
```

S'applique à :

Objet Query

Méthode OrSummaryFilterBy

Ajoute une expression de filtre de récapitulatif à l'interrogation. S'il existe déjà un filtre de récapitulatif, cette expression est ajoutée au filtre à l'aide de l'opérateur OR (OU). Pour en savoir davantage sur la création d'expressions pour le catalogue Impromptu, reportez-vous à la section « Expressions Impromptu dans PowerPrompts » (p. 102).

Le paramètre filtreRécapitulatif correspond à l'expression de filtre de récapitulatif à ajouter.

Paramètres

filtreRécapitulatif : Chaîne, obligatoire

Type de données renvoyées

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet d'ajouter un filtre de récapitulatif avant de renvoyer une instruction SQL.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_didacticiels_js.imr");
myQuery.OrSummaryFilterBy("[\\Produits\\Produit] commencepar 'VA'");
rs.Open(myQuery.SQL, "VAV");
%>
```

S'applique à :

Objet Query

Méthode RemoveColumn

Supprime une colonne dans l'interrogation et le rapport.

Le paramètre nomColonne correspond au nom de la colonne à supprimer de l'interrogation.

Paramètres

nomColonne : Chaîne, obligatoire

Type de données renvoyées

Chaîne

Exemple

L'exemple de syntaxe ci-dessous supprime la colonne *NomFournisseur* de l'interrogation avant de renvoyer l'instruction SQL.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_didacticiels_js.imr");
myQuery.RemoveColumn("NomFournisseur");
rs.Open(myQuery.SQL, "VAV");
%>
```

S'applique à :

Objet Query

Méthode SetDistinct

Indique que l'interrogation contient uniquement des éléments distincts.

Paramètres

onSwitch: Booléen, obligatoire

Type de données renvoyées

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet de supprimer toute valeur en double avant de renvoyer l'instruction SQL.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_didacticiels_js.imr");
myQuery.SetDistinct("vrai");
rs.Open(myQuery.SQL, "VAV");
%>
```

S'applique à :

Objet Query

Méthode SetPromptValue

Définit une valeur pour une demande nommée dans l'interrogation principale. Cette méthode fonctionne uniquement si l'interrogation est créée dans un rapport Impromptu. Toutes les demandes doivent comprendre des valeurs avant de pouvoir extraire l'instruction SQL pour l'interrogation. Pour s'assurer que le rapport Impromptu accepte les valeurs de demandes multiples, utilisez l'opérateur IN (DANS) au lieu du signe égal (=) dans l'instruction associée au filtre pour ce rapport Impromptu.

Conseil : Utilisez une virgule pour séparer les valeurs multiples. Lorsqu'il est nécessaire d'insérer une virgule dans une valeur, faites précéder cette virgule d'un caractère d'échappement. De manière implicite, le caret (^) correspond au caractère d'échappement. Cependant, vous pouvez le modifier en mettant à jour l'entrée Separator Escape Character de la section [Startup Options] du fichier Impromptu.ini.

Paramètres

Le paramètre nomDemande correspond au nom de la demande dans l'interrogation. Utilisez le nom de la demande de la zone *Demandes disponibles* dans la boîte de dialogue *Gestionnaire des demandes*.

Le paramètre valeurDemande correspond à la valeur que vous souhaitez attribuer à la demande.

nomDemande : Chaîne, obligatoire

valeurDemande : Chaîne, obligatoire

Type de données renvoyées

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet de définir la valeur Canada pour la demande Pays avant de renvoyer l'instruction SQL.

```
<%
var rs = new Recordset();
var myQuery = new Query("pwp_didacticiels_js.imr");
myQuery.SetPromptValue("Pays", "Canada");
rs.Open(myQuery.SQL, "VAV");
%>
```

S'applique à :

Objet Query

Méthode SortBy

Outre les tris déjà définis dans l'interrogation, trie l'interrogation en fonction de la colonne spécifiée.

Paramètres

Le paramètre colonneTri correspond au nom de la colonne à trier. Le paramètre ordreTri correspond soit à l'ordre croissant soit à l'ordre décroissant permettant de définir dans quel ordre la colonne doit être triée.

colonneTri : Chaîne, obligatoire

ordreTri : Chaîne (croissant ou décroissant), obligatoire

Type de données renvoyées

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet de trier l'interrogation en fonction de la colonne *TypeProduit* avant de renvoyer l'instruction SQL.

```
<%
var myQuery = new Query("pwp_didacticiels_js.imr");
myQuery.SortBy("TypeProduit","croissant");
rs.Open(myQuery.SQL, "VAV");
%>
```

S'applique à :

Objet Query

Objet Recordset

Extrait les données des bases de données logiques Cognos. Les méthodes associées à l'objet Recordset permettent de se connecter aux bases de données, d'exécuter des instructions SQL et de récupérer les lignes de données.

Il s'agit d'un objet qui peut être créé.

Propriété	Description
Propriété CurrentRecordIndex	Renvoie l'index de l'enregistrement en cours qui doit commencer à 0. Cette valeur est incrémentée chaque fois que la méthode MoveNext est appelée. Cette propriété est valide uniquement si la propriété EOF est définie sur *Faux*.
Propriété EOF	Recherche la fin de l'ensemble des enregistrements.
Propriété Fields	Collection d'objets Field. Chaque zone représente une colonne dans les résultats de l'interrogation.
Propriété MaxRecords	Définit ou renvoie le nombre maximum d'enregistrements à renvoyer.

Méthode	Description
Méthode Close	Ferme les enregistrements ouverts.
Méthode MoveNext	Passe à la ligne suivante dans les enregistrements.
Méthode Open	Insère dans les enregistrements les lignes renvoyées lors de l'exécution de l'instruction SQL.

Propriété CurrentRecordIndex

Renvoie l'index de l'enregistrement en cours qui doit commencer à 0. Cette valeur est incrémentée chaque fois que la méthode MoveNext est appelée. Cette propriété est valide uniquement si la Propriété EOF est définie sur *Faux*.

Cette propriété est en lecture seule.

Type

Entier

Exemple

L'exemple de syntaxe ci-dessous permet d'afficher le nom de chaque pays sur une ligne distincte précédée par le numéro de ligne.

```
<%
var rs = new Recordset();
rs.Open("Select Pays from Pays","VAV");
while (!rs.EOF)
{
    Response.Writeln("Ligne : " + rs.CurrentRecordIndex + " " + rs.Fields("pays"));
    Response.Write("<br>");
    rs.MoveNext();
}
%>
```

Exemple de résultat

```
Ligne : 0 France
Ligne : 1 Allemagne
Ligne : 2 États-Unis
Ligne : 3 Canada
Ligne : 4 Autriche
Ligne : 5 Italie
Ligne : 6 Pays-Bas
Ligne : 7 Suisse
Ligne : 8 Angleterre
Ligne : 9 Suède
Ligne : 10 Japon
Ligne : 11 Taïwan
Ligne : 12 Corée
Ligne : 13 Chine
Ligne : 14 Australie
Ligne : 15 Belgique
Ligne : 16 Danemark
Ligne : 17 Espagne
Ligne : 18 Mexique
Ligne : 19 Brésil
Ligne : 20 Finlande
Ligne : 21 Union européenne
```

S'applique à :

Objet Recordset

Propriété EOF

Recherche la fin de l'ensemble des enregistrements. EOF signifie End Of File (Fin du fichier). Si l'interrogation qui résulte ne contient pas de ligne, alors la valeur du paramètre EOF est défini sur *Vrai*.

Cette propriété est en lecture seule.

Type

Booléen

Exemple

L'exemple de syntaxe ci-dessous permet d'afficher le nom de chaque pays sur une ligne distincte.

```
<%
var rs = new Recordset();
rs.Open("Select CodePaysVente,Pays from Pays", "VAV");
while(!rs.EOF)
{
    Response.Write("<option value='" + rs.Fields("CodePaysVente") + "'>" +
rs.Fields("Pays") + "</option>");
    Response.Write("<br>");
    rs.MoveNext();
}
rs.Close();
%>
```

Exemple de résultat

```
France
Allemagne
États-Unis
Canada
Autriche
Italie
Pays-Bas
Suisse
Angleterre
Suède
Japon
Taïwan
Corée
Chine
Australie
Belgique
Danemark
Espagne
Mexique
Brésil
Finlande
Union européenne
```

S'applique à :

Objet Recordset

Propriété Fields

Collection d'objets Field. Chaque zone représente une colonne dans les résultats de l'interrogation.

Cette propriété est en lecture seule.

Type

Collection

Exemple

```
<%
var rs = new Recordset();
rs.Open("Select * from Pays","VAV");
Response.Write("Nombre de zones dans les résultats : " + rs.Fields.Count);
%>
```

Exemple de résultat

```
Nombre de zones dans les résultats : 7
```

S'applique à :

Objet Recordset

Propriété MaxRecords

Définit ou renvoie le nombre maximum d'enregistrements à renvoyer.

Pour que cette propriété fonctionne correctement, vous devez la définir avant d'appeler la méthode Recordset.Open.

Cette propriété possède un accès lecture-écriture.

Type

Entier

Valeur implicite

0 -- Aucun enregistrement n'est renvoyé

-1 -- Nombre illimité d'enregistrements

Exemple

```
<%
var rs = new Recordset();
rs.MaxRecords = 10;
rs.Open("Select CodePaysVente,Pays from Pays", "VAV");
while(!rs.EOF)
{
    Response.Write("<option value='" + rs.Fields("CodePaysVente") + "'>" +
rs.Fields("Pays") + "</option>");
    Response.Write("<br>");
    rs.MoveNext();
}
%>
```

Exemple de résultat

```
France
Allemagne
États-Unis
Canada
Autriche
Italie
Pays-Bas
Suisse
Angleterre
Suède
```

S'applique à :

Objet Recordset

Méthode Close

Ferme les enregistrements. Lorsque des enregistrements ne sont pas fermés de façon explicite, PowerPrompts les ferme à la fin de la page. Appliquez cette méthode pour réutiliser des objets Recordset.

Paramètres

Aucun

Type de données renvoyées

Aucun

Exemple

L'exemple de syntaxe ci-dessous permet d'afficher le nom de chaque pays, puis de chaque gamme de produits.

```
<%
var rs = new Recordset();
rs.Open("Select CodePaysVente,Pays from Pays", "VAV");
// Les enregistrements doivent être fermés avant de pouvoir réutiliser l'objet Recordset
while(!rs.EOF)
{
    Response.Write("<option value='" + rs.Fields("CodePaysVente") + "'>" +
rs.Fields("Pays") + "</option>");
    Response.Write(", ");
    rs.MoveNext();
}
Response.Write("<br><br>");
rs.Close();
rs.Open("Select GammeProduits from GammeProduits","VAV");
while(!rs.EOF)
{
    Response.Write("<option>" + rs.Fields("GammeProduits") + "</option>");
    Response.Write(", ");
    rs.MoveNext();
}
%>
```

Exemple de résultat

```
France, Allemagne, États-Unis, Canada, Autriche, Italie, Pays-Bas, Suisse, Angleterre,
Suède, Japon, Taïwan, Corée, Chine, Australie, Belgique, Danemark, Espagne, Mexique,
Brésil, Finlande, Union européenne.
Matériel de camping, Matériel de montagne, Accessoires personnels, Articles de
protection, Matériel de golf.
```

S'applique à :

Objet Recordset

Méthode MoveNext

Passe à la ligne suivante dans les enregistrements.

Paramètres

Aucun

Type de données renvoyées

Aucun

Exemple

L'exemple de syntaxe ci-dessous permet d'afficher chaque nom de pays et la devise correspondante sur la même ligne.

```
<%
var rs = new Recordset();
rs.Open("Select NomMonnaie,Pays from Pays order by Pays", "VAV");
while(!rs.EOF)
{
    Response.Writeln(rs.Fields("Pays") + " " + rs.Fields("NomMonnaie"));
    Response.Write("<br>");
    rs.MoveNext();
}
rs.Close();
%>
```

Exemple de résultat

```
Australie dollars
Autriche schillings
Belgique francs
Brésil reals
Canada dollars
Chine renminbis
Danemark couronnes
Angleterre livres
Union européenne euros
Finlande marks
France francs
Allemagne marks
Italie lires
Japon yens
Corée wons
Mexique pesos
Pays-bas florins
Espagne pesetas
Suède couronnes
Suisse francs
Taïwan nouveaux dollars
États-Unis dollars
```

S'applique à :

Objet Recordset

Méthode Open

Insère dans les enregistrements les lignes renvoyées lors de l'exécution de l'instruction SQL.

Paramètres

Le paramètre SQL correspond à l'instruction SQL à exécuter. Le paramètre nomBD correspond au nom de la base de données logique Cognos.

Il existe deux paramètres de chaînes facultatifs. Le paramètre codeUtilisateurBD correspond au code d'utilisateur de base de données à utiliser lors de la connexion à la base de donnée. Le paramètre motdepasseBD correspond au mot de passe de base de données pour ce code d'utilisateur.

SQL : Chaîne, obligatoire

nomBD : Chaîne, obligatoire

codeUtilisateurBD : Chaîne, facultatif

motdepasseBD : Chaîne, facultatif

Type de données renvoyées

Aucun

Exemple 1

L'exemple de syntaxe ci-dessous permet d'ouvrir des enregistrements dans une base de données logique pour laquelle il n'est pas nécessaire de saisir un code d'accès.

```
<%
var rs = new Recordset();
rs.Open("Select CodeGammeProduits, GammeProduits from GammeProduits order by
GammeProduits", "VAV");
while(!rs.EOF)
{
    Response.Write("<option value='" + rs.Fields("CodeGammeProduits") + "'>" +
rs.Fields("GammeProduits") + "</option>");
    Response.Write("<br>");
    rs.MoveNext();
}
rs.Close();
%>
```

Exemple de résultat 1

```
Matériel de camping
Matériel de golf
Matériel de montagne
Articles de protection
Accessoires personnels
```

Exemple 2

L'exemple de syntaxe ci-dessous illustre comment utiliser une instruction SQL dans un rapport Impromptu.

L'exemple de code ci-dessous est identique à celui de l'exemple 1, à l'exception de la ligne 2 (rs.Open…) que vous devez remplacer par les lignes suivantes :

```
var myQuery = new Query("pwp_didacticiels_js.imr");
rs.Open(myQuery.SQL, "VAV");
```

Exemple 3

L'exemple de syntaxe ci-dessous illustre comment utiliser une instruction SQL à partir d'une chaîne.

L'exemple de code ci-dessous est identique à celui de l'exemple 1, à l'exception de la ligne 2 (rs.Open…) que vous devez remplacer par les lignes suivantes :

```
var sSQL = new String;
sSQL = "Select Pays from Pays";
rs.Open(sSQL.SQL, "VAV");
```

S'applique à :

Objet Recordset

Objet Request

Fournit des informations sur la requête en cours du client.

Propriété	Description
Propriété Cookies	Renvoie tous les cookies entrants sous forme de liste de chaînes.
Propriété ServerVariables	Renvoie toutes les variables du serveur sous forme de liste de chaînes.
Propriété Variables	Renvoie les variables de la requête en cours, différentes des variables d'application toujours existantes entre les requêtes, sous forme de liste de chaînes.

Propriété Cookies

Renvoie tous les cookies entrants sous forme de liste de chaînes. Un cookie se constitue d'une paire de valeurs/noms envoyée par le navigateur. Les cookies sont classés en fonction du niveau de détail de leurs attributs de chemins, du plus détaillé au moins détaillé.

Cette propriété est en lecture seule.

Type

Liste de chaînes

Exemple 1

L'exemple de syntaxe ci-dessous permet de répertorier tous les cookies et les valeurs qui y sont associées.

```
<%
// Répertorier tous les cookies
for (var myCookie in Request.Cookies)
{
    Response.Write(myCookie + " = " + Request.Cookies(myCookie) + "<br/>");
}
%>
```

Exemple 2

L'exemple de syntaxe ci-dessous indique la valeur du cookie appelée NomCookie.

```
<% =Request.Cookies("NomCookie") %>
```

S'applique à :

Objet Request

Propriété ServerVariables

Renvoie toutes les variables du serveur sous forme de liste de chaînes. Vous ne pouvez pas indexer les variables du serveur en fonction d'un nombre car il n'existe pas d'ordre pour ces variables.

Les variables de serveur suivantes peuvent être renvoyées :

AUTH_TYPE	AUTH_USER
CONTENT_LENGTH	CONTENT_TYPE
GATEWAY_INTERFACE	HTTP_ACCEPT
HTTP_ACCEPT_LANGUAGE	HTTP_COOKIE
HTTP_USER_AGENT	HTTP_ACCEPT_CHARSET
HTTP_HOST	HTTP_RANGE
HTTP_REFERER	HTTPS
LOGON_USER	PATH_INFO
PATH_TRANSLATED	QUERY_STRING
REMOTE_ADDR	REMOTE_HOST
REMOTE_IDENT	REMOTE_USER
REQUEST_METHOD	SCRIPT_NAME
SERVER_NAME	SERVER_PORT
SERVER_PROTOCOL	SERVER_SOFTWARE

Cette propriété est en lecture seule.

Type

Liste de chaînes

Exemple

L'exemple de syntaxe ci-dessous indique la valeur de la variable du serveur appelé SERVER_NAME.

```
<%=Request.ServerVariables("SERVER_NAME")%>
```

S'applique à :

Objet Request

Propriété Variables

Renvoie les variables de la requête en cours, différentes des variables d'application toujours existantes entre les requêtes, sous forme de liste de chaînes. Utilisez cette propriété uniquement lorsque vous avez besoin de paires de variables de formulaire non traitées. Utilisez la propriété App.Variables pour extraire les variables définies dans l'application.

Cette propriété est en lecture seule.

Type

Liste de chaînes

Exemple

L'exemple de syntaxe ci-dessous permet d'affecter la valeur de la variable CodePays à la valeur de la variable JavaScript myValue. La variable CodePays doit être définie sur une page précédente.

```
<%
var myValue = Request.Variables("CodePays");
%>
```

S'applique à :

Objet Request

Objet Response

Renvoie le contenu au navigateur.

Propriété	Description
Propriété ContentType	Indique le type de contenu HTTP de la réponse.

Méthode	Description
Méthode AppendCookie	Définit un cookie.
Méthode AppendHeader	Ajoute une zone à l'en-tête HTTP de la réponse.
Méthode Clear	Supprime le contenu et les zones d'en-tête HTTP.
Méthode ClearContent	Supprime la totalité du contenu de la réponse.
Méthode ClearHeaders	Supprime toutes les zones d'en-tête HTTP de la réponse.

Méthode	Description
Méthode Redirect	Renvoie le navigateur vers une autre adresse URL.
Méthode Write	Insère une chaîne dans la réponse que le navigateur reçoit.
Méthode WriteFile	Insère le contenu d'un fichier dans la réponse que le navigateur reçoit.
Méthode Writeln	Insère une chaîne (suivie d'une nouvelle ligne) dans la réponse que le navigateur reçoit.

Propriété ContentType

Indique le type de contenu HTTP de la réponse. Pour en savoir davantage sur le type de codage MIME, consultez les normes HTTP.

S'il s'agit d'une page HTML, vous n'avez pas besoin de définir cette propriété car le protocole HTML est le protocole implicite. Définissez cette propriété une seule fois par page.

Cette propriété possède un accès lecture-écriture.

Type

Chaîne

Valeur implicite

Texte/HTML

Exemple 1

L'exemple de syntaxe ci-dessous indique comment envoyer du texte brut au navigateur.

```
<%
Response.ContentType = "texte/texte brut";
Response.Write("Il s'agit d'un message en texte brut.");
%>
```

Exemple de résultat 1

```
Il s'agit d'un message en texte brut.
```

Exemple 2

L'exemple de syntaxe ci-dessous indique comment envoyer un classeur Excel au navigateur.

```
<%
Response.ContentType = "application/vnd.ms-excel";
Response.AppendHeader("Disposition-Contenu", "inline; filename=file1.xls");
%>
<table>
<tr>
<td>Numéro du produit</td>
<td>Nom du produit</td>
<td>Coût</td>
</tr>
<tr>
<td align=left>105</td>
<td>Ensemble de bois Ouragan en titane</td>
<td align=left>555.90$</td>
</tr>
</table>
```

S'applique à :

Objet Response

Méthode AppendCookie

Définit un cookie. Pour en savoir davantage sur les cookies, consultez les spécifications sur les cookies de Netscape.

Paramètres

nomCookie : Chaîne, obligatoire

Exemple

```
<% Response.AppendCookie( "MyCookie=Laventus3; path=/" ), %>
```

S'applique à :

Objet Response

Méthode AppendHeader

Ajoute une zone à l'en-tête HTML de la réponse. Cette méthode constitue une fonction avancée et n'est généralement pas nécessaire.

Paramètres

ZoneEn-tête: Chaîne, obligatoire

Exemple

```
<%Response.AppendHeader("Contenu-Type: texte/texte brut");%>
```

S'applique à :

Objet Response

Méthode Clear

Supprime le contenu et les zones d'en-tête HTTP. Lorsque vous utilisez cette méthode, vous obtenez les mêmes résultats que lorsque vous faites appel aux méthodes Response.Clear-Content et Response.ClearHeader. Cette méthode constitue une fonction avancée et n'est généralement pas nécessaire.

Exemple

```
<% Response.Clear(); %>
```

S'applique à :

Objet Response

Méthode ClearContent

Supprime la totalité du contenu de la réponse. Cette méthode constitue une fonction avancée et n'est généralement pas nécessaire.

Exemple

```
<%Response.ClearContent();%>
```

S'applique à :

Objet Response

Méthode ClearHeaders

Supprime toutes les zones d'en-tête HTTP de la réponse. Cette méthode constitue une fonction avancée et n'est généralement pas.

Exemple

```
<%Response.ClearHeaders()%>
```

S'applique à :

Objet Response

Méthode Redirect

Renvoie le navigateur vers une autre adresse URL.

Paramètres

URL : Chaîne, obligatoire

Exemple

L'exemple de syntaxe ci-dessous permet de renvoyer le navigateur vers le site Web Cognos.

```
<%Response.Redirect( "www.cognos.com" );%>
```

S'applique à :

Objet Response

Méthode Write

Insère une chaîne dans la réponse que le navigateur reçoit. Cette méthode convertit automatiquement les données numériques en chaînes de caractères.

Paramètres

uneChaîne : Chaîne, obligatoire

Type de données renvoyées

Aucun

Exemple 1

L'exemple de syntaxe ci-dessous permet d'envoyer du texte au navigateur.

```
<%
// Afficher une chaîne dans le navigateur
Response.Write("<b>Cette chaîne est en gras.</b>");
%>
```

Exemple 2

L'exemple de syntaxe ci-dessous permet d'envoyer une liste déroulante de produits au navigateur.

```
<select name="ProdType">
<%
var rs = new Recordset;
rs.Open("Select CodeTypeProduit, TypeProduit from TypeProduit where CodeGammeProduits =
" + App.Variables("GammeProduits"), "VAV");
while(!rs.EOF)
{
    Response.Write("<option value='" + rs.Fields("CodeTypeProduit") + "'>" +
rs.Fields("TypeProduit") + "</option>");
    rs.MoveNext();
}
rs.Close();
%>
</select>
```

Exemple de résultat

```
Fers
Bois
Fers droits
Accessoires de golf
```

S'applique à :

Objet Response

Méthode WriteFile

Insère le contenu d'un fichier dans la réponse que le navigateur reçoit.

Le moteur JavaScript côté serveur n'interprète pas ce type de texte.

Paramètres

nomFichier : Chaîne, obligatoire

Exemple

L'exemple de syntaxe ci-dessous permet d'afficher le contenu du fichier CompanyPage-Footer.txt dans le navigateur.

```
<% Response.WriteFile( App.Path + "CompanyPageFooter.txt" ); %>
```

S'applique à :

Objet Response

Méthode WriteIn

Insère une chaîne (suivie d'une nouvelle ligne) dans la réponse que le navigateur reçoit.

Remarque : Si le contenu est de type HTML et que vous souhaitez insérer une nouvelle ligne dans le navigateur, utilisez la balise
.

Paramètres

uneChaîne : Chaîne, obligatoire

Type de données renvoyées

Aucun

Exemple 1

L'exemple de syntaxe ci-dessous permet d'afficher les noms de chaque pays ainsi que les devises correspondantes sur une même ligne.

```
<%
var rs = new Recordset();
rs.Open("Select NomMonnaie,Pays from Pays order by Pays", "VAV");
while(!rs.EOF)
{
    Response.Writeln(rs.Fields("Pays") + " " + rs.Fields("NomMonnaie"));
    Response.Write("<br>");
    rs.MoveNext();
}
rs.Close();
%>
```

Exemple de résultat 1

```
Australie dollars
Autriche schillings
Belgique francs
Brésil reals
Canada dollars
Chine renminbis
Danemark couronnes
Angleterre livres
Union européenne euros
Finlande marks
France francs
Allemagne marks
Italie lires
Japon yens
Corée wons
Mexique pesos
Pays-bas florins
Espagne pesetas
Suède couronnes
Suisse francs
Taïwan nouveaux dollars
États-Unis dollars
```

Exemple 2

L'exemple de syntaxe ci-dessous permet d'afficher la balise Body sur une nouvelle ligne dans le navigateur.

```
Response.Writeln('<BODY BGCOLOR="#FFFFFF"><table width="100%" bgcolor="#cddfff"
cellpadding="0" cellspacing="0"><tr><td height="1" bgcolor="#003366"
width="100%"></td></tr><tr><td><H1>Détails d'un représentant</H1></td></tr></table>')
```

S'applique à :

Objet Response

Objet Server

Permet d'accéder aux méthodes et propriétés sur le serveur. La plupart de ces méthodes et de ces propriétés sont utilisées comme fonctions d'utilitaires.

Propriété	Description
Propriété ScriptTimeout	Définit ou renvoie le nombre de secondes pendant lesquelles le serveur doit attendre avant de renvoyer l'erreur indiquant que le délai d'exécution du script a été atteint.

Méthode	Description
Méthode CreateObject	Crée l'instance de l'objet COM (Component Object Model) pour un IDprog. Fonctionne uniquement sous Windows.
Méthode FormatNumber	Applique un format à un nombre.
Méthode HTMLEncode	Applique un code HTML à une chaîne spécifiée.

Méthode	Description
Méthode URLDecode	Applique les règles de décodage pour les adresses URL. Il s'agit de la méthode inverse à la méthode URLEncode.
Méthode URLEncode	Applique les règles de codage pour les adresses URL. Il s'agit de la méthode inverse à la méthode URLDecode.

Propriété ScriptTimeout

Définit ou renvoie le nombre de secondes pendant lesquelles le serveur doit attendre avant de renvoyer l'erreur indiquant que le délai d'exécution du script a été atteint. Utilisez cette propriété pour éviter qu'une boucle infinie ne soit générée sur le serveur.

Cette propriété possède un accès lecture-écriture.

Type

Entier

Valeur implicite

90

Exemple

```
<%
// Renvoie la page d'erreur après 20 secondes
Server.ScriptTimeout = 20;
var i = 0;
while (true)
{
    i++
}
%>
```

Exemple de résultat

```
Erreur
La valeur Server.ScriptTimeout (20 secondes) a été dépassée pendant l'exécution de la
ligne numéro de la ligne à la page nom de la page.
i++
Pour en savoir davantage, consultez le journal des événements du serveur.
```

S'applique à :

Objet Server

Méthode CreateObject

Crée l'instance de l'objet COM (Component Object Model) pour un IDprog. Cette méthode fonctionne uniquement sous Windows car il n'existe pas de prise en charge de l'objet COM sous UNIX.

Paramètres

IDprog : Chaîne

Type de données renvoyées

Objet

Exemple

L'exemple de syntaxe ci-dessous indique comment extraire des données à l'aide du composant ADO. Pour en savoir davantage sur la technologie ADO, consultez la documentation de Microsoft.

```
<%
    var ocom = Server.CreateObject("ADODB.Connection");
// Toute entrée ODBC DSN valide
oConn.Open("GOScer3");
// Toute valeur SQL pour le DSN ci-dessus
var oRs = oConn.Execute("Select Pays,CodePaysVente from Pays order by Pays");
Response.Writeln("<select name='country_cd'>");
while (!oRS.EOF)
{
    Response.Writeln("<option value='" + oRs.Fields("CodePaysVente").Value + "'>" +
oRs.Fields("Pays").Value + "</option>");
    oRs.MoveNext();
}
Response.Writeln("</select>");
%>
```

S'applique à :

Objet Server

Méthode FormatNumber

Applique un format à un nombre. Pour en savoir davantage sur les formats numériques dans Impromptu, consultez la rubrique *Symboles de format numérique* dans l'aide en ligne d'Impromptu à l'intention des utilisateurs.

Paramètres

Nombre : Chaîne, obligatoire

Format : Chaîne, obligatoire

Type de données renvoyées

Chaîne

Exemple

```
<%
var myNumber = "67.0000012";
Response.Write(Server.FormatNumber(myNumber,"#,#.00"));
%>
```

Exemple de résultat

```
67.00
```

S'applique à :

Objet Server

Méthode HTMLEncode

Applique un code HTML à une chaîne spécifiée. Par exemple, cette méthode permet de remplacer tous les signes inférieurs à (<) par <, tous les signes supérieurs à (>) par > et l'esperluette (&) par &. Pour en savoir davantage sur le code HTML, consultez votre documentation HTML.

Paramètres

Valeur : Chaîne

Type de données renvoyées

Chaîne

Exemple

```
<%
var myString;
myString = "<>&'\"";
Response.Write(Server.HTMLEncode(myString));
%>
```

Exemple de résultat

```
<>&'"
```

Séquence du navigateur

```
&lt;&gt;&amp;&apos;&quot;
```

S'applique à :

Objet Server

Méthode URLDecode

Applique les règles de décodage pour les adresses URL. Par exemple, cette méthode permet de remplacer les valeurs hexadécimales par les caractères équivalents. Il s'agit de la méthode inverse à la méthode URLEncode.

Paramètres

Valeur : Chaîne

Type de données renvoyées

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet de coder, puis de décoder une chaîne.

```
<%
var myString;
myString= " &^*@";
var myString2;
myString2 = ""
Response.Write(Server.URLEncode(myString));
Response.Write("<br>")
myString2 = Server.URLEncode(myString);
Response.Write(Server.URLDecode(myString2));
%>
```

Exemple de résultat

```
%20%26%5E%2A%40
&^*@
```

S'applique à :

Objet Server

Méthode URLEncode

Applique les règles de codage pour les adresses URL. Par exemple, cette méthode permet de remplacer les caractères non alphanumériques par les valeurs hexadécimales équivalentes. Il s'agit de la méthode inverse à la méthode URLDecode.

Paramètres

Valeur : Chaîne

Type de données renvoyées

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet de coder, puis de décoder une chaîne.

```
<%
var myString;
myString= " &^*@";
var myString2;
myString2 = ""
Response.Write(Server.URLEncode(myString));
Response.Write("<br>")
myString2 = Server.URLEncode(myString);
Response.Write(Server.URLDecode(myString2));
%>
```

Exemple de résultat

```
%20%26%5E%2A%40
&^*@
```

S'applique à :

Objet Server

Objet StringList

Collection de valeurs de chaînes.

Propriété	Description
Propriété Count	Renvoie le nombre de valeurs de chaînes dans la liste de chaînes.

Méthode	Description
Méthode Contains	Vérifie si la liste de chaînes contient une valeur.
Méthode Item	Extrait une valeur de chaîne indexée.
Méthode Join	Renvoie toutes les valeurs concaténées. Vous avez la possibilité de spécifier un séparateur. Le caractère implicite est la virgule (,).
Méthode Operator() (Liste de chaînes)	Indique l'opérateur d'index utilisé pour accéder aux éléments de la collection. Vous pouvez spécifier un index numérique.
Méthode toString	Renvoie toutes les valeurs concaténées. Cette méthode est identique à la propriété Join, mais elle ne requiert pas de paramètres.

Propriété Count

Renvoie le nombre de valeurs de chaînes dans la liste de chaînes.

Cette propriété est en lecture seule.

Type

Entier

Exemple

L'exemple de syntaxe ci-dessous permet d'ajouter une liste déroulante de pays dans une page, puis dans la page suivante, d'imprimer le nombre de pays que vous avez sélectionnés.

Page 1

```
<select name="Countries" multiple>
<%
var rs = new Recordset;
rs.Open("Select Pays from Pays order by Pays", "VAV");
while(!rs.EOF)
{
    Response.Write("<option>" + rs.Fields("Pays") + "</option>");
    rs.MoveNext();
}
rs.Close();
%>
</select>
```

Page 2

```
<%
var myStringList = App.Variables("Pays");
Response.Write("Il existe " + myStringList.Count + " éléments de chaîne.");
%>
```

Exemple de résultat

```
Il existe 22 éléments de chaîne.
```

S'applique à :

Objet StringList

Méthode Contains

Vérifie si la liste de chaînes contient une valeur.

Paramètres

ValeuràRechercher : Chaîne, obligatoire

Type de données renvoyées

Booléen

Exemple

L'exemple de syntaxe ci-dessous permet d'ajouter une liste déroulante de pays dans une page, puis dans la page suivante, de vérifier si le pays Canada est l'un des pays que vous avez sélectionné.

Page 1

```
<select name="Countries" multiple>
<%
var rs = new Recordset;
rs.Open("Select Pays from Pays order by Pays", "VAV");
while(!rs.EOF)
{
    Response.Write("<option>" + rs.Fields("Pays") + "</option>");
    rs.MoveNext();
}
rs.Close();
%>
</select>
```

Page 2

```
<%
var myStringList = App.Variables("Pays");
// Renseigner l'objet StringList
if (myStringList.Contains("Canada"))
{
    Response.Write("Le Canada a été sélectionné.");
}
else
{
    Response.Write("Le Canada n'a pas été sélectionné.");
}
%>
```

Exemple de résultat

```
Le Canada a été sélectionné.
```

S'applique à :

Objet StringList

Méthode Item

Extrait une valeur de chaîne indexée. L'index commence à 0.

Paramètres

Index : Entier, obligatoire

Type de données renvoyées

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet d'ajouter une liste déroulante de pays dans une page, puis, dans la page suivante, d'afficher le deuxième élément de la chaîne.

Page 1

```
<select name="Countries" multiple>
<%
var rs = new Recordset;
rs.Open("Select Pays from Pays order by Pays", "VAV");
while(!rs.EOF)
{
    Response.Write("<option>" + rs.Fields("Pays") + "</option>");
    rs.MoveNext();
}
rs.Close();
%>
</select>
```

Page 2

```
<%
var myStringList = App.Variables("Pays");
var x = new String();
x = myStringList.Item(1);
// L'index commençant à 0, il s'agit donc du second élément affiché
Response.Write(x + " est le second élément de la chaîne.");
%>
```

Exemple de résultat

```
Autriche est le second élément de la chaîne.
```

S'applique à :

Objet StringList

Méthode Join

Renvoie toutes les valeurs concaténées. Vous avez la possibilité de spécifier un séparateur. Le caractère implicite est la virgule (,).

Paramètres

Séparateur : Chaîne, obligatoire

Type de données renvoyées

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet d'ajouter une liste déroulante de pays dans une page, puis, dans la page suivante, d'afficher les noms des pays que vous avez sélectionnés, séparés par une virgule.

Page 1

```
<select name="Countries" multiple>
<%
var rs = new Recordset;
rs.Open("Select Pays from Pays order by Pays", "VAV");
while(!rs.EOF)
{
    Response.Write("<option>" + rs.Fields("Pays") + "</option>");
    rs.MoveNext();
}
rs.Close();
%>
</select>
```

Page 2

```
<%
var myStringList = App.Variables("Pays").Join(", ");
Response.Write(myStringList);
%>
```

Exemple de résultat

```
Australie, Autriche, Belgique
```

S'applique à :

Objet StringList

Méthode Operator() (Liste de chaînes)

Indique l'opérateur d'index utilisé pour accéder aux éléments de la collection. L'index commence à 0.

Paramètres

Vous pouvez spécifier un index numérique.

Index : Numérique

Type de données renvoyées

Élément de la collection

Exemple

L'exemple de syntaxe ci-dessous permet de renvoyer le premier pays indiqué dans la liste de chaîne.

```
<%=App.Variables("Pays")(0)%>
```

S'applique à :

Objet StringList

Méthode toString

Renvoie toutes les valeurs concaténées. Cette méthode est identique à la propriété Join, mais elle ne requiert pas de paramètres.

Il s'agit du paramètre implicite de la liste de chaînes.

Paramètres

Aucun

Type de données renvoyées

Chaîne

Exemple

```
<%
switch(App.Variables("MoisCommande").toString())
{
    case "1" :
        Response.Write("<tr><td>Mois Commande</td><td>" + "Janvier" + "</td></tr>");
        break;
...
}
%>
```

S'applique à :

Objet StringList

Objet Upfront

Permet d'effectuer une intégration avec le serveur Upfront afin que les applications PowerPrompts puissent utiliser des thèmes et des langues.

Propriété	Description
Propriété Language	Renvoie la langue actuellement sélectionnée dans Upfront.
Propriété Locale	Renvoie le paramètre régional actuellement sélectionné dans Upfront.
Propriété Theme	Renvoie le nom du thème actuellement sélectionné dans Upfront.

Méthode	Description
Méthode ExecuteCommand	Permet de transmettre une commande XML au serveur Upfront. La valeur renvoyée correspond au document XML indiquant la réussite ou l'échec de la commande ainsi que les résultats.
Méthode GetPageFragment	Renvoie la partie d'une page HTML dans Upfront en fonction du thème et de la langue sélectionnés. La valeur renvoyée correspond au contenu de la page tel qu'il a été traité par le module de modèles Upfront.

Propriété Language

Renvoie la langue actuellement sélectionnée dans Upfront.

Cette propriété est en lecture seule.

Type

Chaîne

Exemple

```
<%
// Indiquer à l'utilisateur que l'application est disponible en anglais uniquement
if (Upfront.Language != "en")
{
    Response.Write("<b>Vous avez sélectionné une langue autre que l'anglais. Cette
application est disponible en anglais uniquement.</b>");
}
else
{
    Response.Write("<b>Upfront est en anglais.</b>");
}
%>
```

Exemple de résultat

```
Upfront est en anglais.
```

S'applique à :

Objet Upfront

Propriété Locale

Renvoie le paramètre régional actuellement sélectionné dans Upfront.

Cette propriété est en lecture seule.

Type

Chaîne

Exemple

```
<%
// Indiquer à l'utilisateur le paramètre régional actuellement sélectionné
Response.Write("Paramètre régional : " + Upfront.Locale);
%>
```

Exemple de résultat

```
Paramètre régional : en-us
```

S'applique à :

Objet Upfront

Propriété Theme

Renvoie le nom du thème actuellement sélectionné dans Upfront.

Cette propriété est en lecture seule.

Type

Chaîne

Exemple

```
<!-- Indiquer à l'utilisateur le thème actuellement sélectionné -->
Thème : <%=Upfront.Theme%>
```

Exemple de résultat

```
Thème : standard70
```

S'applique à :

Objet Upfront

Méthode ExecuteCommand

Permet de transmettre une commande XML au serveur Upfront. La valeur renvoyée correspond au document XML indiquant la réussite ou l'échec de la commande ainsi que les résultats.

Vous devez spécifier l'espace-noms approprié pour l'élément racine de la commande. L'espace-noms est http://developer.cognos.com/schemas/upfront.

Paramètres

CommandeXML : Chaîne, obligatoire

Type de données renvoyées

Chaîne

Exemple

```
<%=Server.HTMLEncode(Upfront.ExecuteCommand("<DescribeUser
xmlns=\"http://developer.cognos.com/schemas/upfront/\"/>"))%>
```

S'applique à :

Objet Upfront

Méthode GetPageFragment

Renvoie la partie d'une page HTML dans Upfront en fonction du thème et de la langue sélectionnés. La valeur renvoyée correspond au contenu de la page tel qu'il a été traité par le module de modèles Upfront.

Paramètres

NomSectionPage : Chaîne, obligatoire

Type de données renvoyées

Chaîne

Exemple

```
<%=Upfront.GetPageFragment("header.html")%>
```

S'applique à :

Objet Upfront

Objet User

Fournit des informations sur l'utilisateur pour un permis valide dans Access Manager.

Propriété	Description
Propriété Description	Renvoie la description de l'utilisateur actuellement défini dans Access Manager.
Propriété Email	Renvoie l'adresse électronique de l'utilisateur actuellement défini dans Access Manager.

Propriété	Description
Propriété Telephone	Renvoie le numéro de téléphone de l'utilisateur actuellement défini dans Access Manager.
Propriété UserClass	Renvoie le nom de la classe d'utilisateurs de l'utilisateur actuel. La propriété UserClass correspond à la classe d'utilisateurs définie pour l'exécution du rapport.
Propriété UserClasses	Répertorie les classes d'utilisateurs disponibles pour cet utilisateur. La propriété UserClasses correspond à la liste de toutes les classes d'utilisateurs auxquelles l'utilisateur appartient actuellement dans Access Manager.
Propriété UserName	Renvoie le nom de l'utilisateur actuellement défini dans Access Manager.

Propriété Description

Renvoie la description de l'utilisateur actuellement défini dans Access Manager.

Cette propriété est en lecture seule.

Type

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet d'indiquer le nom de l'utilisateur, la classe d'utilisateurs, son adresse électronique, son numéro de téléphone et une description de l'utilisateur sous forme de tableau.

```
<%
// Afficher uniquement les propriétés de l'utilisateur sous forme de tableau
Response.Writeln("<table>");
Response.Writeln("<tr><td>NomUtilisateur :</td><td>" + User.UserName + "</td></tr>");
Response.Writeln("<tr><td>ClasseUtilisateurs :</td><td>" + User.UserClass +
"</td></tr>");
Response.Writeln("<tr><td>CourrierÉlectronique :</td><td>" + User.Email + "</td></tr>");
Response.Writeln("<tr><td>Téléphone :</td><td>" + User.Telephone + "</td></tr>");
Response.Writeln("<tr><td>Description :</td><td>" + User.Description + "</td></tr>");
Response.Writeln("</table>");
%>
```

S'applique à :

Objet User

Propriété Email

Renvoie l'adresse électronique de l'utilisateur actuellement défini dans Access Manager.

Cette propriété est en lecture seule.

Type

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet d'indiquer le nom de l'utilisateur, la classe d'utilisateurs, son adresse électronique, son numéro de téléphone et une description de l'utilisateur sous forme de tableau.

```
<%
// Afficher uniquement les propriétés de l'utilisateur sous forme de tableau
Response.Writeln("<table>");
Response.Writeln("<tr><td>NomUtilisateur :</td><td>" + User.UserName + "</td></tr>");
Response.Writeln("<tr><td>ClasseUtilisateurs :</td><td>" + User.UserClass +
"</td></tr>");
Response.Writeln("<tr><td>CourrierÉlectronique :</td><td>" + User.Email + "</td></tr>");
Response.Writeln("<tr><td>Téléphone :</td><td>" + User.Telephone + "</td></tr>");
Response.Writeln("<tr><td>Description :</td><td>" + User.Description + "</td></tr>");
Response.Writeln("</table>");
%>
```

S'applique à :

Objet User

Propriété Telephone

Renvoie le numéro de téléphone de l'utilisateur actuellement défini dans Access Manager.

Cette propriété est en lecture seule.

Type

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet d'indiquer le nom de l'utilisateur, la classe d'utilisateurs, son adresse électronique, son numéro de téléphone et une description de l'utilisateur sous forme de tableau.

```
<%
// Afficher uniquement les propriétés de l'utilisateur sous forme de tableau
Response.Writeln("<table>");
Response.Writeln("<tr><td>NomUtilisateur :</td><td>" + User.UserName + "</td></tr>");
Response.Writeln("<tr><td>ClasseUtilisateurs :</td><td>" + User.UserClass +
"</td></tr>");
Response.Writeln("<tr><td>CourrierÉlectronique :</td><td>" + User.Email + "</td></tr>");
Response.Writeln("<tr><td>Téléphone :</td><td>" + User.Telephone + "</td></tr>");
Response.Writeln("<tr><td>Description :</td><td>" + User.Description + "</td></tr>");
Response.Writeln("</table>");
%>
```

S'applique à :

Objet User

Propriété UserClass

Renvoie le nom de la classe d'utilisateurs de l'utilisateur actuel. La propriété UserClass correspond à la classe d'utilisateurs définie pour l'exécution du rapport.

Cette propriété est en lecture seule.

Type

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet d'indiquer le nom de l'utilisateur, la classe d'utilisateurs, son adresse électronique, son numéro de téléphone et une description de l'utilisateur sous forme de tableau.

```
<%
// Afficher uniquement les propriétés de l'utilisateur sous forme de tableau
Response.Writeln("<table>");
Response.Writeln("<tr><td>NomUtilisateur :</td><td>" + User.UserName + "</td></tr>");
Response.Writeln("<tr><td>ClasseUtilisateurs :</td><td>" + User.UserClass +
"</td></tr>");
Response.Writeln("<tr><td>CourrierÉlectronique :</td><td>" + User.Email + "</td></tr>");
Response.Writeln("<tr><td>Téléphone :</td><td>" + User.Telephone + "</td></tr>");
Response.Writeln("<tr><td>Description :</td><td>" + User.Description + "</td></tr>");
Response.Writeln("</table>");
%>
```

S'applique à :

Objet User

Propriété UserClasses

Répertorie les classes d'utilisateurs disponibles pour cet utilisateur. La propriété UserClasses correspond à la liste de toutes les classes d'utilisateurs auxquelles l'utilisateur appartient actuellement dans Access Manager.

Cette propriété est en lecture seule.

Type

Liste de chaînes

Exemple

L'exemple de syntaxe ci-dessous permet de lier toutes les classes d'utilisateurs disponibles pour cet utilisateur.

```
<%=UserClasses.Join()%>
```

S'applique à :

Objet User

Propriété UserName

Renvoie le nom de l'utilisateur actuellement défini dans Access Manager.

Cette propriété est en lecture seule.

Type

Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet d'indiquer le nom de l'utilisateur, la classe d'utilisateurs, son adresse électronique, son numéro de téléphone et une description de l'utilisateur sous forme de tableau.

```
<%
// Afficher uniquement les propriétés de l'utilisateur sous forme de tableau
Response.Writeln("<table>");
Response.Writeln("<tr><td>NomUtilisateur :</td><td>" + User.UserName + "</td></tr>");
Response.Writeln("<tr><td>ClasseUtilisateurs :</td><td>" + User.UserClass +
"</td></tr>");
Response.Writeln("<tr><td>CourrierÉlectronique :</td><td>" + User.Email + "</td></tr>");
Response.Writeln("<tr><td>Téléphone :</td><td>" + User.Telephone + "</td></tr>");
Response.Writeln("<tr><td>Description :</td><td>" + User.Description + "</td></tr>");
Response.Writeln("</table>");
%>
```

S'applique à :

Objet User

Méthodes de modification Javascript pour les rapports

Utilisez les méthodes de modification JavaScript pour les rapports Impromptu. Utilisez la boîte de dialogue *Éditeur de script* pour ajouter ces méthodes à vos applications PowerPrompts. Vous ne pouvez pas utiliser ces méthodes dans vos pages HTML. Pour en savoir davantage sur un script, reportez-vous à la section « Script : aperçu » (p. 23).

Règles de syntaxe

- Les méthodes associées aux rapports doivent être précédées de la méthode GetReport().
- Les méthodes associées aux objets de rapports doivent être précédées de la méthode GetReportObject().
- Les méthodes associées colonnes doivent être précédées de la méthode GetColumnTitle().
- Les méthodes associées aux interrogations doivent être précédées de la méthode GetQuery().
- Le langage JavaScript fait la distinction entre les majuscules et les minuscules. Par exemple, AddDataItem est correct, mais ADDdataITEM ne l'est pas.
- Placez les expressions Impromptu que vous utilisez avec ces méthodes entre guillemets doubles.
- Placez les valeurs de chaîne dans les expressions Impromptu entre guillemets simples.

  Pour obtenir des exemples appliquant ces règles, reportez-vous à la section « Ajout d'un script » (p. 23).

Remarque

Pour accéder à l'aide en ligne sur une méthode, sélectionnez le nom de la méthode, puis appuyez sur F1 dans la boîte de dialogue *Éditeur de script*.

Méthode GetColumnTitle

Renvoie le titre de la colonne spécifiée.

Syntaxe

```
GetColumnTitle(nomColonne)
```

Paramètres

nomColonne : Chaîne

Exemple

L'exemple de la syntaxe ci-dessous permet de remplacer le titre de colonne *Produit* par le titre *Nom du produit*.

```
GetColumnTitle("Produit").SetText("Nom du produit");
```

Documentation complémentaire

- « Méthode ApplyStyle pour les titres de colonnes » (p. 96)
- « Méthode GetQuery » (p. 89)
- « Méthode GetReport » (p. 90)
- « Méthode GetReportObject » (p. 90)
- « Méthode SetText pour les titres de colonnes » (p. 96)
- « Méthode SetTextJustification pour les titres de colonnes » (p. 97)

Méthode GetQuery

Renvoie l'interrogation principale du rapport.

Syntaxe

```
GetQuery()
```

Exemple

```
GetQuery().AddColumn("Quantité totale","total([Qté])");
```

Documentation complémentaire

- « Méthode AddColumn pour les interrogations » (p. 97)
- « Méthode AndFilterBy pour les interrogations » (p. 98)
- « Méthode AssociateColumn pour les interrogations » (p. 99)
- « Méthode GetColumnTitle » (p. 89)
- « Méthode GetReport » (p. 90)
- « Méthode GetReportObject » (p. 90)
- « Méthode GroupBy pour les interrogations » (p. 99)
- « Méthode SetMaxRows pour les interrogations » (p. 100)

Méthode GetReport

Renvoie le rapport ayant le nom qui a été spécifié.

Syntaxe

```
GetReport()
```

Exemple

L'exemple de syntaxe ci-dessous permet d'ajouter la colonne *Qté* du catalogue au rapport final.

```
GetReport().AddDataItem("Qté","[\\Commandes\\Détails de la commande\\Qté]");
```

Documentation complémentaire

- « Méthode ApplyTemplate pour les rapports » (p. 91)
- « Méthode GetColumnTitle » (p. 89)
- « Méthode GetQuery » (p. 89)
- « Méthode GetReportObject » (p. 90)
- « Méthode RemoveReportObject pour les rapports » (p. 92)
- « Méthode SetListInsertCursor pour les rapports » (p. 92)
- « Méthode SetPrimaryFrame pour les rapports » (p. 93)

Méthode GetReportObject

Renvoie l'objet de rapport dont le nom a été spécifié.

Syntaxe

```
GetReportObject(nomObjRap)
```

Paramètres

nomObjRap : Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet d'appliquer le style Rouge vif à la colonne *Col1*.

```
GetReportObject("Col1").ApplyStyle("Rouge vif");
```

Documentation complémentaire

- « Méthode ApplyStyle pour les objets de rapports » (p. 94)
- « Méthode GetColumnTitle » (p. 89)
- « Méthode GetQuery » (p. 89)
- « Méthode GetReport » (p. 90)
- « Méthode SetText pour les objets de rapports » (p. 95)
- « Méthode VerticalAlign pour les objets de rapports » (p. 96)

Méthode AddDataItem pour les rapports

Ajoute une expression spécifique sous forme de colonne dans l'interrogation et insère l'objet de rapport correspondant. L'objet est inséré à la position implicite du cadre principal.

Syntaxe

```
AddDataItem(nomObjRap,données)
```

Paramètres

nomObjRap : Chaîne

données : Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet d'ajouter la colonne *Qté* du catalogue au rapport final.

```
GetReport().AddDataItem("Qté","[\\Commandes\\Détails de la commande\\Qté]");
```

Documentation complémentaire

- « Méthode ApplyTemplate pour les rapports » (p. 91)
- « Méthode GetReport » (p. 90)
- « Méthode GetReportObject » (p. 90)
- « Méthode RemoveReportObject pour les rapports » (p. 92)
- « Méthode SetListInsertCursor pour les rapports » (p. 92)
- « Méthode SetPrimaryFrame pour les rapports » (p. 93)

Méthode ApplyTemplate pour les rapports

Applique le modèle spécifié au rapport.

Syntaxe

```
ApplyTemplate(nomModèle)
```

Paramètres

nomModèle : Chaîne

Exemple 1

L'exemple de syntaxe ci-dessous permet d'appliquer le modèle Laventus.imt au rapport. Ce modèle se trouve sous C:\Applications PowerPrompts.

```
GetReport().ApplyTemplate("C:\\Applications PowerPrompts\\Laventus.imt");
```

Exemple 2

L'exemple de syntaxe ci-dessous permet d'appliquer le modèle Laventus.imt au rapport. Le modèle Laventus.imt se trouve dans le dossier Modèles sous le dossier contenant l'application PowerPrompts.

```
GetReport().ApplyTemplate(".\\Modèles\\Laventus.imt");
```

Remarque : Vous devez toujours spécifier le nom du modèle mais vous pouvez spécifier un chemin d'accès relatif ou absolu. Les chemins d'accès relatifs se réfèrent à l'emplacement des rapports. Vous pouvez obtenir le chemin d'accès de l'application en appliquant la propriété App.Path.

Documentation complémentaire

- « Méthode AddDataItem pour les rapports » (p. 91)
- « Méthode GetReport » (p. 90)
- « Méthode RemoveReportObject pour les rapports » (p. 92)
- « Méthode SetListInsertCursor pour les rapports » (p. 92)
- « Méthode SetPrimaryFrame pour les rapports » (p. 93)

Méthode RemoveReportObject pour les rapports

Supprime l'objet de rapport nommé du rapport.

Cette méthode ne fonctionne pas pour les colonnes masquées dans les rapports Impromptu.

Syntaxe

```
RemoveReportObject(nomObjRap)
```

Paramètres

nomObjRap : Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet de supprimer l'objet de rapport nommé *Pays* du rapport final.

```
GetReport().RemoveReportObject("Pays");
```

Documentation complémentaire

- « Méthode AddDataItem pour les rapports » (p. 91)
- « Méthode ApplyTemplate pour les rapports » (p. 91)
- « Méthode GetReport » (p. 90)
- « Méthode SetListInsertCursor pour les rapports » (p. 92)
- « Méthode SetPrimaryFrame pour les rapports » (p. 93)

Méthode SetListInsertCursor pour les rapports

Définit l'emplacement où les colonnes sont ajoutés dans les listes. De manière implicite, les nouvelles colonnes sont ajoutées à droite des colonnes existantes.

Spécification	Place les nouvelles colonnes
-1	à droite des colonnes existantes
0	à gauche des colonnes existantes
1	à droite de la première colonne existante
2	à droite de la seconde colonne existante
N (où N correspond à un nombre entier positif)	à droite de la Nième colonne existante

Syntaxe

```
SetListInsertCursor(IndexInsertionColonne)
```

Paramètres

IndexInsertionColonne : Entier

Exemple

```
GetReport().SetListInsertCursor(0);
```

Remarque : Dans la boîte de dialogue *Éditeur de script*, cette méthode doit apparaître avant la méthode AddDataItem car le script doit d'abord définir le point d'insertion des éléments avant que les colonnes puissent être ajoutées.

Documentation complémentaire

- « Méthode AddDataItem pour les rapports » (p. 91)
- « Méthode ApplyTemplate pour les rapports » (p. 91)
- « Méthode GetReport » (p. 90)
- « Méthode RemoveReportObject pour les rapports » (p. 92)
- « Méthode SetPrimaryFrame pour les rapports » (p. 93)

Méthode SetPrimaryFrame pour les rapports

Définit le cadre principal du rapport. Le cadre principal correspond à l'emplacement dans lequel Impromptu ajoute des objets de rapport.

Syntaxe

```
SetPrimaryFrame(nomCadre)
```

Paramètres

nomCadre : Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet de définir le cartouche *Type de produit* comme cadre principal pour le rapport.

```
GetReport().SetPrimaryFrame("Cartouche Type de produit");
```

Documentation complémentaire

- « Méthode AddDataItem pour les rapports » (p. 91)
- « Méthode ApplyTemplate pour les rapports » (p. 91)
- « Méthode GetReport » (p. 90)
- « Méthode RemoveReportObject pour les rapports » (p. 92)
- « Méthode SetListInsertCursor pour les rapports » (p. 92)

Méthode AddConditionalFormat pour les objets de rapports

Ajoute un objet de mise en forme conditionnelle à l'objet de rapport. Une mise en forme conditionnelle est constituée d'une condition nommée et d'un style nommé.

Syntaxe

```
AddConditionalFormat(conditionNommée,nomStyle)
```

Paramètres

conditionNommée : Chaîne

nomStyle : Chaîne

Exemple

L'exemple de syntaxe ci-dessous permet d'ajouter une mise en forme conditionnelle à l'objet Qty.

```
GetReportObject("Qté").AddConditionalFormat("Petit montant","Mauvais");
```

Documentation complémentaire

- « Méthode ApplyStyle pour les objets de rapports » (p. 94)
- « Méthode GetReportObject » (p. 90)
- « Méthode HorizontalAlign pour les objets de rapports » (p. 94)
- « Méthode HorizontalAlignToColumn pour les objets de rapports » (p. 95)
- « Méthode SetText pour les objets de rapports » (p. 95)
- « Méthode SetTextJustification pour les objets de rapports » (p. 95)
- « Méthode VerticalAlign pour les objets de rapports » (p. 96)

Méthode ApplyStyle pour les objets de rapports

Applique le style nommé à l'objet de rapport. Le style doit être indiqué dans le fichier Impromptu.ini sur le serveur où l'application PowerPrompts est installée.

Syntaxe

```
ApplyStyle(nomStyle)
```

Paramètres

nomStyle : Chaîne

Exemple

```
GetReportObject("Produit").ApplyStyle("Rouge 1 visible");
```

Documentation complémentaire

- « Méthode AddConditionalFormat pour les objets de rapports » (p. 93)
- « Méthode GetReportObject » (p. 90)
- « Méthode HorizontalAlign pour les objets de rapports » (p. 94)
- « Méthode HorizontalAlignToColumn pour les objets de rapports » (p. 95)
- « Méthode SetText pour les objets de rapports » (p. 95)
- « Méthode SetTextJustification pour les objets de rapports » (p. 95)
- « Méthode VerticalAlign pour les objets de rapports » (p. 96)

Méthode HorizontalAlign pour les objets de rapports

Aligne horizontalement l'objet de rapport en fonction du parent.

Syntaxe

```
HorizontalAlign(typeAlignement)
```

Paramètres

typeAlignement : Chaîne (à gauche, centré ou à droite)

Exemple

```
GetReportObject("Titre").HorizontalAlign("centré");
```

Documentation complémentaire

- « Méthode AddConditionalFormat pour les objets de rapports » (p. 93)
- « Méthode ApplyStyle pour les objets de rapports » (p. 94)
- « Méthode GetReportObject » (p. 90)
- « Méthode HorizontalAlignToColumn pour les objets de rapports » (p. 95)
- « Méthode SetText pour les objets de rapports » (p. 95)
- « Méthode SetTextJustification pour les objets de rapports » (p. 95)
- « Méthode VerticalAlign pour les objets de rapports » (p. 96)

Méthode HorizontalAlignToColumn pour les objets de rapports

Aligne horizontalement l'objet de rapport en fonction de la colonne spécifiée. Par exemple, vous pouvez utiliser cette méthode pour les récapitulatifs se trouvant dans un cartouche.

Syntaxe

```
HorizontalAlignToColumn(alignerAuNomColonne,typeAlignement)
```

Paramètres

alignerAuNomColonne: Chaîne

typeAlignement : Chaîne (à gauche, centré ou à droite)

Exemple

```
GetReportObject("Quantité totale").HorizontalAlignToColumn("Qté","à droite");
```

Documentation complémentaire

- « Méthode AddConditionalFormat pour les objets de rapports » (p. 93)
- « Méthode ApplyStyle pour les objets de rapports » (p. 94)
- « Méthode GetReportObject » (p. 90)
- « Méthode HorizontalAlign pour les objets de rapports » (p. 94)
- « Méthode SetText pour les objets de rapports » (p. 95)
- « Méthode SetTextJustification pour les objets de rapports » (p. 95)
- « Méthode VerticalAlign pour les objets de rapports » (p. 96)

Méthode SetText pour les objets de rapports

Définit le texte d'un cadre.

Syntaxe

```
SetText(texte)
```

Paramètres

texte : Chaîne

Exemple

```
GetReportObject("Titre").SetText("État financier");
```

Documentation complémentaire

- « Méthode AddConditionalFormat pour les objets de rapports » (p. 93)
- « Méthode ApplyStyle pour les objets de rapports » (p. 94)
- « Méthode GetReportObject » (p. 90)
- « Méthode HorizontalAlign pour les objets de rapports » (p. 94)
- « Méthode HorizontalAlignToColumn pour les objets de rapports » (p. 95)
- « Méthode SetTextJustification pour les objets de rapports » (p. 95)
- « Méthode VerticalAlign pour les objets de rapports » (p. 96)

Méthode SetTextJustification pour les objets de rapports

Aligne le texte dans le cadre.

Syntaxe

```
SetTextJustification(typeJustification)
```

Paramètres

typeJustification : Chaîne (à gauche, centré ou à droite)

Exemple

```
GetReportObject("Produit").SetTextJustification("à droite");
```

Documentation complémentaire

- « Méthode AddConditionalFormat pour les objets de rapports » (p. 93)
- « Méthode ApplyStyle pour les objets de rapports » (p. 94)
- « Méthode GetReportObject » (p. 90)
- « Méthode HorizontalAlign pour les objets de rapports » (p. 94)
- « Méthode HorizontalAlignToColumn pour les objets de rapports » (p. 95)
- « Méthode SetText pour les objets de rapports » (p. 95)
- « Méthode VerticalAlign pour les objets de rapports » (p. 96)

Méthode VerticalAlign pour les objets de rapports

Aligne verticalement l'objet de rapport en fonction de son cadre englobant.

Syntaxe

```
VerticalAlign(typeAlignement)
```

Paramètres

typeAlignement : Chaîne (en haut, centré ou en bas)

Exemple

```
GetReportObject("Quantité totale").VerticalAlign("centré");
```

Documentation complémentaire

- « Méthode AddConditionalFormat pour les objets de rapports » (p. 93)
- « Méthode ApplyStyle pour les objets de rapports » (p. 94)
- « Méthode GetReportObject » (p. 90)
- « Méthode HorizontalAlign pour les objets de rapports » (p. 94)
- « Méthode HorizontalAlignToColumn pour les objets de rapports » (p. 95)
- « Méthode SetText pour les objets de rapports » (p. 95)
- « Méthode SetTextJustification pour les objets de rapports » (p. 95)

Méthode ApplyStyle pour les titres de colonnes

Applique le style nommé au titre de colonne.

Syntaxe

```
ApplyStyle(nomStyle)
```

Paramètres

nomStyle : Chaîne

Exemple

```
GetColumnTitle("Produit").ApplyStyle("Rouge 1 visible");
```

Documentation complémentaire

- « Méthode GetColumnTitle » (p. 89)
- « Méthode SetText pour les titres de colonnes » (p. 96)
- « Méthode SetTextJustification pour les titres de colonnes » (p. 97)

Méthode SetText pour les titres de colonnes

Définit le texte d'un titre de colonne.

Syntaxe

```
SetText(texte)
```

Paramètres

texte : Chaîne

Exemple

```
GetColumnTitle("Type de produit").SetText("Type du produit");
```

Documentation complémentaire

- « Méthode ApplyStyle pour les titres de colonnes » (p. 96)
- « Méthode GetColumnTitle » (p. 89)
- « Méthode SetTextJustification pour les titres de colonnes » (p. 97)

Méthode SetTextJustification pour les titres de colonnes

Aligne le texte dans le cadre.

Syntaxe

```
SetTextJustification(typeJustification)
```

Paramètres

typeJustification : Chaîne (à gauche, centré ou à droite)

Exemple

```
GetColumnTitle("Produit").SetTextJustification("à droite");
```

Documentation complémentaire

- « Méthode ApplyStyle pour les titres de colonnes » (p. 96)
- « Méthode GetColumnTitle » (p. 89)
- « Méthode SetText pour les titres de colonnes » (p. 96)

Méthode AddColumn pour les interrogations

Ajoute une colonne à l'interrogation pour laquelle le nom et l'expression ont été spécifiés.

Syntaxe

```
AddColumn(nomCol,données)
```

Paramètres

nomCol : Chaîne

données : Chaîne

Exemple

```
GetQuery().AddColumn("Quantité totale","total([Qté])");
```

Documentation complémentaire

- « Méthode AssociateColumn pour les interrogations » (p. 99)
- « Méthode GetQuery » (p. 89)
- « Expressions Impromptu dans PowerPrompts » (p. 102)
- « Méthode RemoveColumn pour les interrogations » (p. 100)

Méthode AddNamedCondition pour les interrogations

Ajoute une condition nommée à l'interrogation. Utilisez cette méthode pour définir une mise en forme conditionnelle.

Syntaxe

```
AddNamedCondition(nomCondition,condition)
```

Paramètres

nomCondition : Chaîne

condition : Chaîne

Exemple

```
GetQuery().AddNamedCondition("Ventes médiocres","[\\Produits\\Historique des
ventes\\Ventes 95] < 5000");
```

Documentation complémentaire

- « Méthode GetQuery » (p. 89)

Méthode AndFilterBy pour les interrogations

Ajoute une condition spécifiée au filtre défini. Si le filtre défini n'est pas vide, la condition est ajoutée à l'aide de l'opérateur AND (ET).

Remarque : Cette méthode ne fonctionne pas pour les rapports Impromptu dans lesquels vous avez défini l'instruction SQL pour l'interrogation.

Syntaxe

```
AndFilterBy(condition)
```

Paramètres

condition : Chaîne

Exemple

```
GetQuery().AndFilterBy("année([DateCommande]) in (95,96)");
```

Documentation complémentaire

- « Méthode AndSummaryFilterBy pour les interrogations » (p. 98)
- « Méthode GetQuery » (p. 89)
- « Méthode OrFilterBy pour les interrogations » (p. 99)

Méthode AndSummaryFilterBy pour les interrogations

Ajoute une condition spécifiée au filtre de récapitulatif défini. Si le filtre défini n'est pas vide, la condition est ajoutée à l'aide de l'opérateur AND (ET).

Remarque : Cette méthode ne fonctionne pas pour les rapports Impromptu dans lesquels vous avez défini l'instruction SQL pour l'interrogation.

Syntaxe

```
AndSummaryFilterBy(condition)
```

Paramètres

condition : Chaîne

Exemple

```
GetQuery().AndSummaryFilterBy("[Ventes totales] > 10000");
```

Documentation complémentaire

- « Méthode AndFilterBy pour les interrogations » (p. 98)
- « Méthode GetQuery » (p. 89)
- « Méthode OrSummaryFilterBy pour les interrogations » (p. 100)

Méthode AssociateColumn pour les interrogations

Associe la colonne spécifiée à la colonne de groupe spécifiée.

Syntaxe

```
AssociateColumn(nomCol,nomColGroup)
```

Paramètres

nomCol : Chaîne

nomColGroup : String

Exemple

```
GetQuery().AssociateColumn("Ventes totales","Pays");
```

Documentation complémentaire

- « Méthode AddColumn pour les interrogations » (p. 97)
- « Méthode GetQuery » (p. 89)
- « Méthode RemoveColumn pour les interrogations » (p. 100)

Méthode GroupBy pour les interrogations

Ajoute la colonne nommée à la fin de la liste de colonnes groupées. La colonne est triée dans l'ordre croissant ou décroissant si l'ordre de tri a été défini respectivement sur croissant et décroissant.

Syntaxe

```
GroupBy(nomCol,typeTri)
```

Paramètres

nomCol : Chaîne

typeTri : Chaîne (croissant ou décroissant)

Exemple

```
GetQuery().GroupBy("Client","décroissant");
```

Documentation complémentaire

- « Méthode AddColumn pour les interrogations » (p. 97)
- « Méthode GetQuery » (p. 89)
- « Méthode SortBy pour les interrogations » (p. 101)

Méthode OrFilterBy pour les interrogations

Ajoute une condition spécifiée au filtre défini. Si le filtre défini n'est pas vide, la condition est ajoutée à l'aide de l'opérateur OR (OU).

Remarque : Cette méthode ne fonctionne pas pour les rapports Impromptu dans lesquels vous avez défini l'instruction SQL pour l'interrogation.

Syntaxe

```
OrFilterBy(condition)
```

Paramètres

condition : Chaîne

Exemple

```
GetQuery().OrFilterBy("[\\Produits\\Produit] commencepar('VA')");
```

Documentation complémentaire

- « Méthode AndFilterBy pour les interrogations » (p. 98)
- « Méthode GetQuery » (p. 89)
- « Méthode OrSummaryFilterBy pour les interrogations » (p. 100)

Méthode OrSummaryFilterBy pour les interrogations

Ajoute une condition spécifiée au filtre de récapitulatif défini. Si le filtre défini n'est pas vide, la condition est ajoutée à l'aide de l'opérateur OR (OU).

Remarque : Cette méthode ne fonctionne pas pour les rapports Impromptu dans lesquels vous avez défini l'instruction SQL pour l'interrogation.

Syntaxe

```
OrSummaryFilterBy(condition)
```

Paramètres

condition : Chaîne

Exemple

```
GetQuery().OrSummaryFilterBy("[Pays] = 'Canada'");
```

Documentation complémentaire

- « Méthode AndSummaryFilterBy pour les interrogations » (p. 98)
- « Méthode GetQuery » (p. 89)
- « Méthode OrFilterBy pour les interrogations » (p. 99)

Méthode RemoveColumn pour les interrogations

Supprime la colonne spécifiée de l'interrogation et supprime tous les objets de rapports qui font référence à la colonne.

Syntaxe

```
RemoveColumn(nomCol)
```

Paramètres

nomCol : Chaîne

Exemple

```
GetQuery().RemoveColumn("Type de produit");
```

Documentation complémentaire

- « Méthode AddColumn pour les interrogations » (p. 97)
- « Méthode GetQuery » (p. 89)
- « Méthode AssociateColumn pour les interrogations » (p. 99)

Méthode SetMaxRows pour les interrogations

Définit le nombre maximum de lignes que l'interrogation peut renvoyer dans les résultats.

Syntaxe

```
SetMaxRows(lignesMax)
```

Paramètres

lignesMax : Entier

Exemple

```
GetQuery().SetMaxRows(3000);
```

Documentation complémentaire

- « Méthode GetQuery » (p. 89)

Méthode SetPromptValue pour les interrogations

Définit les valeurs de la demande dans le rapport Impromptu associé à l'application PowerPrompts. Pour s'assurer que le rapport Impromptu accepte les valeurs de demandes multiples, utilisez l'opérateur IN au lieu du signe égal (=) dans l'instruction associée au filtre pour ce rapport.

Conseil : Utilisez une virgule pour séparer les valeurs. Lorsqu'il est nécessaire d'insérer une virgule dans une valeur, faites précéder cette virgule d'un caractère d'échappement. De manière implicite, le caret (^) correspond au caractère d'échappement. Cependant, vous pouvez le modifier en mettant à jour l'entrée Separator Escape Character de la section [Startup Options] du fichier Impromptu.ini.

Syntaxe

```
SetPromptValue(nomDemande,valeur1,...,valeurN)
```

Paramètres

Pour le paramètre nomDemande, utilisez le nom de la demande indiqué dans la zone *Demandes disponibles* de la boîte de dialogue *Gestionnaire des demandes*.

nomDemande : Chaîne

valeur1-valeurN : Variant

Exemples

Dans l'exemple ci-dessous, "Date filtre" correspond au nom de la demande et "28-02-1999" à la valeur de la demande.

```
GetQuery().SetPromptValue("Date filtre","28-02-1999")
GetQuery().SetPromptValue("Code Pays","App.Variables("maVar")");
```

Documentation complémentaire

- « Méthode GetQuery » (p. 89)

Méthode SortBy pour les interrogations

Ajoute la colonne nommée à la fin de la liste des colonnes triées. La colonne est triée dans l'ordre croissant ou décroissant si l'ordre de tri a été défini respectivement sur croissant et décroissant.

Syntaxe

```
SortBy(nomCol,typeTri)
```

Paramètres

nomCol : Chaîne

typeTri : Chaîne (croissant ou décroissant)

Exemple

```
GetQuery().SortBy("Pays","croissant")
```

Documentation complémentaire

- « Méthode AddColumn pour les interrogations » (p. 97)
- « Méthode GetQuery » (p. 89)
- « Méthode GroupBy pour les interrogations » (p. 99)

Expressions Impromptu dans PowerPrompts

Les expressions Impromptu modifie une interrogation et sont utilisées pour les calculs, les filtres et les conditions. Les expressions Impromptu ressemblent à celles de PowerPrompts mais elles ne sont pas identiques.

Dans PowerPrompts, vous saisissez directement des expressions Impromptu dans la boîte de dialogue *Éditeur de script*, mais le code inséré correspond à du code JavaScript. Si vous souhaitez créer des expressions Impromptu complexes, créez-les, dans un premier temps, dans Impromptu (à l'aide d'un éditeur d'expressions), puis recréez-les dans PowerPrompts en y insérant les modifications JavaScript suivantes :

- Supprimez les espaces dans les noms des fonctions, par exemple, « commence par » doit être « commencepar ».
- Utilisez des traits de soulignement et non des tirets pour les noms de fonctions. Par exemple 'date-en-chaîne' doit être 'date_en_chaîne'.
- Faites apparaître les références de catalogue entre crochets. Vous devez également utiliser des doubles barres obliques inverses devant chaque référence. Par exemple, [\\Admin.\\Pays\\Code Pays].
- Faites la distinction entre les références des interrogations et les références de catalogue. Pour cela, n'insérez pas de barre oblique devant les références des interrogations, tel que [Col 1].
- Placez un point d'interrogation avant et après les demandes dans les rapports. Par exemple, ?nomDemande?.
- Désignez les demandes enregistrées dans le catalogue comme vous le feriez pour toute autre colonne de catalogue.

Le tableau suivant indique les deux expressions Impromptu et leurs équivalents dans PowerPrompt.

Expression PowerPrompt	Expression Impromptu
[\Commandes\Date de la commande] >= 01-01-1996	Date de la commande >= 01-01-1996
[\Date] commencepar '1999'	Date commence par '1996'

Lorsque vous insérez l'un des caractères ci-dessous dans une expression Impromptu, vous devez faire précéder ce caractère d'une barre oblique inverse :

- guillemet simple (ou apostrophe) pour un libellé de chaîne,
- barre oblique inverse pour un libellé de chaîne,
- crochet pour une référence de catalogue.

Remarques

- L'utilisation des majuscules et des minuscules n'a aucune importance pour les noms de fonctions. Ainsi, vous pouvez aussi bien saisir exemple 'commencepar' que 'CommencePar'.
- Les espaces entre les opérateurs ne sont pas nécessaires. Ainsi, vous pouvez aussi bien saisir 'x+3*abs(y)' que 'x + 3 * abs (y) '.
- Les constantes numériques doivent commencer par un chiffre. Par exemple vous devez saisir '0.3' au lieu de '.3'.

Pour en savoir davantage sur les expressions Impromptu, reportez-vous à la section *Référence* de l'aide en ligne d'Impromptu.

Méthodes JavaScript Client

Les méthodes JavaScript Client sont fournies avec la bibliothèque JavaScript Client.

Vous pouvez utiliser les méthodes suivantes uniquement pour vos pages HTML.

- Méthode FormatUserVar
- Méthode GetUserVar
- Méthode GetUserVarValues

N'insérez pas ces méthode entre crochets en chevrons (<%...%>) car il s'agit de méthodes JavaScript côté client.

Procédure

- Pour utiliser ces méthodes, vous devez insérer le code suivant dans vos pages HTML :

```
<script language= "javascript"
src="/cognos/PowerPrompts/PowerPromptLib.js"></script>
```

Méthode FormatUserVar

Cette méthode permet de mettre en forme les valeurs d'une variable en fonction de la chaîne de mise en forme spécifiée et du séparateur de liste. L'utilisateur doit définir cette variable lors de la création de l'application PowerPrompts. Le signe du pourcentage (%) sert de caractère de substitution. Pour représenter un pourcentage (%), vous devez saisir deux fois le signe du pourcentage (%%).

Syntaxe

```
FormatUserVar(nomVar,format,séparateur)
```

Paramètres

nomVar : Chaîne

format : Chaîne

séparateur : Chaîne

Exemple

```
<html>
<head>
<script language="javascript" src="/cognos/PowerPrompts/PowerPromptLib.js"></script>
</head>
<body>
<form>
</form>
Indiquez-moi les valeurs dans Pays :
<script language="javascript">
document.write(FormatUserVar("ListeÉtats","état = '%'"," AND "));
</script>
</body>
</html>
```

renvoie :

```
état = 'AK' AND état = 'VT' AND état = 'AL'
```

Documentation complémentaire

- « Méthode GetUserVar » (p. 103)
- « Méthode GetUserVarValues » (p. 104)
- « Méthodes JavaScript Client » (p. 102)

Méthode GetUserVar

Renvoie la valeur d'une variable que l'utilisateur a définie lors de la création de l'application PowerPrompts. S'il existe plusieurs valeurs associées à la variable, ces valeurs sont séparées par une virgule.

Syntaxe

```
GetUserVar(nomVar)
```

Paramètres

nomVar : Chaîne

Exemple

```
<html>
<head>
<script language="javascript" src="/cognos/PowerPrompts/PowerPromptLib.js"></script>
</head>
<body>
<form>
</form>
Indiquez-moi les valeurs dans Pays :
<script language="javascript">
document.write(GetUserVar("pays"));
</script>
</body>
</html>
```

Documentation complémentaire

- « Méthode FormatUserVar » (p. 103)
- « Méthode GetUserVarValues » (p. 104)
- « Méthodes JavaScript Client » (p. 102)

Méthode GetUserVarValues

Renvoie un tableau de valeurs pour la variable nommée.

Syntaxe

```
GetUserVarValues(nomVar)
```

Paramètres

nomVar : Chaîne

Exemple

```
<html>
<head>
<script language="javascript" src="/cognos/PowerPrompts/PowerPromptLib.js"></script>
</head>
<body>
<form>
</form>
Indiquez-moi les valeurs dans Pays :
<script language="javascript">
document.write(GetUserVarValues("code pays"));
</script>
</body>
</html>
```

Documentation complémentaire

- « Méthode FormatUserVar » (p. 103)
- « Méthode GetUserVar » (p. 103)
- « Méthodes JavaScript Client » (p. 102)

Index

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W