What's New

Appeon PowerBuilder® 2019

Contents

1 New Features in PowerBuilder 2019

About this chapter

This chapter introduces the new features in PowerBuilder 2019.

1.1 Empowering PowerBuilder with C# features

.NET Packages

Two .NET packages (**PowerBuilder.Data** and **PowerBuilder.Data.AspNetCore**) are provided to implement the .NET replacement for the PowerBuilder DataStore. **PowerBuilder.Data** provides a pure .NET DataStore compatible with the .NET Core along with related libraries to integrate with PowerBuilder DataWindows or DataStores through a REST interface. The .NET DataStore works similar to the PowerBuilder DataStore and maintains same naming convention of its APIs for easy porting of existing project assets. **PowerBuilder.Data** also provides a ModelStore object which works against the .NET data model and can be used to replace the .NET DataStore.

Also, you can enhance your .NET projects with a new .NET ORM framework (called **SnapObjects**), which is also compatible with the .NET Core just like **PowerBuilder.Data** and **PowerBuilder.Data.AspNetCore**.

For more information about the PowerBuilder .NET APIs and SnapObjects .NET APIs, refer to the Documentation Center.

C# Migration

A batch DataWindow/DataStore conversion utility called **C# Model Generator** is provided to generate the C# data objects and models for the .NET DataStore and ModelStore. For how to use **C# Model Generator**, refer to Part VIII, “Working with C# Model Generator” in *Users Guide*.

Also provided is a SqlExecutorExtention object in **PowerBuilder.Data.AspNetCore**, which can be used to virtually copy n’ paste embedded SQL from PowerScript projects to new C# projects.

C# IDE

A relatively full-featured C# IDE is provided on a standalone basis (called **SnapDevelop (PB Edition)**) and can be launched from the PowerBuilder IDE. It supports development of non-visual projects, such as C# Web APIs, C# non-visual assemblies, and unit testing (with xUnit). And it provides powerful developer productivity tools, such as project wizards, advanced auto-scripting, and C# language services.

For more information about SnapDevelop (PB Edition), refer to the Documentation Center.

1.2 New UI theme

A new UI theming system is provided to allow for codeless approach to how your application UI is rendered. For detailed instructions on how to use the new UI theme, refer to Section 4.4.5, “Specifying a theme for the application UI” in *Users Guide*.

System themes and custom themes

Four new system themes (Flat Design Blue, Flat Design Dark, Flat Design Grey, and Flat Design Silver) are provided for you to apply to the window, DataWindow, and all visual controls (except Line, Oval, Rectangle, RoundRectangle, Picture, PictureHyperLink, and Animation) in your applications. You can also customize these system themes or create your own themes based on these system themes.

Applying a theme

To apply a theme to an application, you can either set it in the Themes tab in the Additional Properties of an application object, or use the ApplyTheme function to set the theme dynamically.

The theme will work in runtime, and has no effect in design time.

Modifying the settings of a theme

If after applying a theme, you want to further adjust the display of certain controls/states, you can open the "theme.json" file of the theme in the specified directory or the default "%Appeon%\Shared\PowerBuilder\theme[version]" directory, and change the corresponding theme settings (with caution).

In case you want to restore the settings of a system theme to its original state, you can do it using the Restore button provided in the Themes tab in the Additional Properties of an application object. The Restore button is only effective when the system theme is located in the default directory.

What can be set by a theme

Due to technical difficulty, a few controls and their states won't reach desired UI effects even through you apply a theme. For more information on what can be set by a theme and what cannot, refer to Section 4.4.5, "Specifying a theme for the application UI" in *Users Guide*.

1.3 Incorporating TX Text Control as built-in editor

Starting from PowerBuilder 2019, a special OEM version of TX Text Control ActiveX is incorporated as a built-in rich text editor in PowerBuilder. This is provided in all editions of PowerBuilder at no additional cost, and is highly recommended to be used for backwards compatibility reasons by all existing PowerBuilder projects that already make use of the RichTextEdit of SAP PowerBuilder version 12.6 or older.

To select to use the OEM version of TX Text Control: in the Application properties dialog box, select the RichTextEdit Control tab, and then select the first option "Built-in TX Text Control".

1.4 New or enhanced PowerBuilder objects

1.4.1 Enhanced RESTClient object

The following functions are added to the RESTClient object:

- Submit can not only send data from the application client to the RESTful web service but also get the response body from the RESTful web service.

  The data to be submitted may be from either a DataWindow, DataStore, DataWindowChild, or JSONPackage. You can specify one or multiple DataWindow

buffers, and also the range in the DataWindow, from which to submit the data. The request supports OAuth 2.0 authorization.

```
objectname.Submit(string urlName, ref string response, DWControl dwObject{,
 boolean format})
```

```
objectname.Submit(string urlName, ref string response, DWControl dwObject
 {,DWBuffer dwbuffer}, boolean changedonly, boolean format)
```

```
objectname.Submit(string urlName, ref string response, DWControl dwObject,
 boolean primarydata, boolean filterdata, boolean deletedata, boolean dwcdata {,
 boolean format})
```

```
objectname.Submit(string urlName, ref string response, DWControl dwObject,
 DWBuffer dwbuffer{,long startrow{, long endrow{, long startcol{, long endcol}}}}
 {, boolean format})
```

```
objectname.Submit(string urlName, ref string response, ref JsonPackage package)
```

- SendDeleteRequest: Sends the HTTP DELETE request to the server and then gets the content of the server response.

  Previously only the HTTPClient object supports sending a request to the RESTful web service. Now you can directly send a request from the RESTClient object, and the request supports OAuth 2.0 authorization.

  ```
  SendDeleteRequest(string urlName{, string data }, ref string response)
  ```

- SendGetRequest: Sends the HTTP GET request to the server and then gets the content of the server response.

  ```
  SendGetRequest(string urlName, ref string response)
  ```

- SendPatchRequest: Sends the HTTP PATCH request to the server and then gets the content of the server response.

  ```
  SendPatchRequest(string urlName, string data, ref string response)
  ```

- SendPostRequest: Sends the HTTP POST request to the server and then gets the content of the server response.

  ```
  SendPostRequest(string urlName, string data, ref string response)
  ```

- SendPutRequest: Sends the HTTP PUT request to the server and then gets the content of the server response.

  ```
  SendPutRequest(string urlName, string data, ref string response)
  ```

- GetJWTToken: Gets the JWT token using the POST method.

  ```
  GetJWTToken (string urlName, string data, ref string token)
  ```

- SetJWTToken: Sets the JWT token string to the HTTP request header which will be sent to the server.

  ```
  SetJWTToken(string jwtToken)
  ```

- GetOAuthToken: Gets the OAuth 2.0 access token.

  ```
  GetOAuthToken (TokenRequest tokenRequest, ref string token)
  ```

- SetOAuthToken: Sets the OAuth 2.0 token string to the HTTP request header which will be sent to the server.

```
SetOAuthToken(string token)
```

- RetrieveOne: Retrieves one data row to the DataWindow, DataWindowChild, or DataStore from the RESTFul Web service.

```
RetrieveOne (DWControl dwObject, string urlName {,string data})
```

The following functions for the RESTClient object are modified:

- Retrieve - Retrieves data to the DataWindow, DataWindowChild, or DataStore from the RESTFul Web service.

  If the data received from the RESTful web service is compressed as gzip, it will be automatically decompressed. Only gzip compression format is supported at this moment. You can use the SetRequestHeader function to set the Accept-Encoding header to allow only the gzip compression format.

- SetRequestHeader - Supports to add a request header or add/replace the value in the existing request header if the header already exists.

```
SetRequestHeader ( string headerName, string headerValue{, Boolean replace } )
```

For more details about these functions, refer to Section 2.85, “RESTClient object” in *Objects and Controls*.

1.4.2 Enhanced HTTPClient object

The following functions for the HTTPClient object are enhanced:

- SetRequestHeader: supports to add a request header or add/replace the value in the existing request header if the header already exists.

```
SetRequestHeader ( string headerName, string headerValue{, Boolean replace } )
```

- SendRequest: supports to encode the data with the charset which is specified by the user in the Content-Type request header, if charset is not specified, this function will encode the data in UTF-8 by default.

- GetResponseBody: supports to encode the data with the charset which is specified by the user in the Content-Type request header; if charset is not specified, this function determines the encoding type based on the BOM header, and then converts the data into UNICODE.

For more details about these functions, refer to Section 2.39, “HTTPClient object” in *Objects and Controls*.

1.4.3 Enhanced JSONPackage object

The GetValue function of the JSONPackage object always returns the result in string, and the SetValue function only sets string-type values. Now there are more functions in the JSONPackage object for getting or setting values of various types.

- GetValueBlob -- Gets the blob value of the key.
- GetValueBoolean -- Gets the boolean value of the key.
- GetValueDate -- Gets the date value of the key.
- GetValueDateTime -- Gets the datetime value of the key.
- GetValueNumber -- Gets the number value of the key.
- GetValueString -- Gets the string value of the key.
- GetValueTime -- Gets the time value of the key.
- SetValueBlob -- Sets the blob value for a key.
- SetValueBoolean -- Sets the boolean value for a key.
- SetValueDate -- Sets the date value for a key.
- SetValueDateTime -- Sets the datetime value for a key.
- SetValueNumber -- Sets the number value for a key.
- SetValueString -- Sets the string value for a key.
- SetValueTime -- Sets the time value for a key.

You can now directly get values from the JSONPackage object into DataWindows, or set the key values in the JSONPackage object from DataWindows.

- GetValueToDataWindow -- Gets the value of the key and inserts it into a DataWindow control, DataStore object, or DataWindowChild object.
- SetValueByDataWindow -- Sets the value of the key using the data from a DataWindow control, DataStore object, or DataWindowChild object.

The JSONPackage object also provides the GetItemType function which works the same as the GetItemType function in the JSONParser object.

- GetItemType -- Gets the type of item.

The SaveToFile and GetJsonBlob functions for JSONPackage (and JSONGenerator) are enhanced to specify the character encoding of the resulting blob.

- SaveToFile

```
SaveToFile ( FileName {, Encoding e} )
```

- GetJsonBlob

```
GetJsonBlob ( {Encoding e} )
```

For the newly-added or enhanced functions, see Section 2.45, “JSONPackage object” in *Objects and Controls* for more details.

The following new property is added to the JSONPackage object (also added to JSONParser):

- ReturnsNullWhenError -- Specifies whether the getting value function returns a null value when error occurs.

  You can use this property to avoid throwing an exception in cases when a getting item function returns null.

  See Section 3.219, “ReturnsNullWhenError” in *Objects and Controls* for more details.

1.4.4 Enhanced JSONGenerator

The SaveToFile and GetJsonBlob functions for JSONGenerator (and JSONPackage) are enhanced to specify the character encoding of the resulting blob.

- SaveToFile

```
SaveToFile ( FileName {, Encoding e} )
```

- GetJsonBlob

```
GetJsonBlob ( {Encoding e} )
```

For more information, see Section 10.554, “SaveToFile” in *PowerScript Reference* and Section 10.239, “GetJsonBlob” in *PowerScript Reference*.

1.4.5 Enhanced JSONParser object

The following new function is added to the JSONParser object:

- ContainsKey -- Checks if the key name exists.

  You can use this function to check whether certain key exists in a JSONParser object before executing other functions, such as GetItem.

  See Section 10.83, “ContainsKey” in *PowerScript Reference* for more details.

The following function is enhanced:

- GetItemType -- Gets the type of item.

  It is now possible to specify the key of a child item, and directly get the type of the child item.

  See Section 10.237, “GetItemType” in *PowerScript Reference* for more details.

The following new property is added to the JSONParser object (also added to JSONPackage):

- ReturnsNullWhenError -- Specifies whether the getting item function returns a null value when error occurs.

  You can use this property to avoid throwing an exception in cases when a getting item function returns null.

  See Section 3.219, “ReturnsNullWhenError” in *Objects and Controls* for more details.

1.4.6 New CompressorObject and ExtractorObject objects

Two new objects called CompressorObject and ExtractorObject are added to compress and decompress the folder or file(s), or the byte data stream. The supported compression formats include ZIP, 7ZIP, GZIP and TAR, and the ZIP and 7ZIP formats support AEM-256 encryption for password; the supported extraction formats include ZIP, 7ZIP, RAR, GZIP, TAR, LZMA, and LZMA86.

For more information about these objects, refer to Section 2.9, “CompressorObject object” in *Objects and Controls* and Section 2.31, “ExtractorObject object” in *Objects and Controls*.

For syntax of compressing files or data stream, refer to Section 10.77, “Compress” in *PowerScript Reference*; for syntax of extracting a compressed archive or data stream, refer to Section 10.141, “Extract” in *PowerScript Reference*; for code examples on compressing and extracting files with the HTTPClient object, the RESTClient object, or the OAuthClient object, refer to Section 18.3, “Compressing and extracting data” in *Application Techniques*.

1.4.7 New Import & Export JSON functions

The following functions are added to import/export a single data row between a JSON string and a DataWindow control, DataStore object, or DataWindowChild object:

- ImportRowFromJson: Inserts a data row from a JSON string into a DataWindow control, DataStore object, or DataWindowChild object.

```
ImportRowFromJson( string json, long row {, ref string error} {, DWBuffer
 dwbuffer})
```

- ExportRowAsJson: Exports a data row from a DataWindow control, DataStore object, or DataWindowChild object to the JSON string.

```
ExportRowAsJson (long row {, DWBuffer dwbuffer})
```

For more information, see Section 9.99, “ImportRowFromJson” in *DataWindow Reference* and Section 9.30, “ExportRowAsJson” in *DataWindow Reference*.

1.4.8 New JSON format

The SnapObjects ModelStore can exchange data with PowerBuilder DataWindow in JSON strings, so now the following JSON formats are supported:

- Plain JSON (formerly called simple JSON)
- DataWindow JSON (formerly called standard JSON)
- ModelStore JSON (newly added in version 2019)

For more about these formats, refer to Section 18.1, “Supported JSON formats” in *Application Techniques*.

1.5 New Windows 10 style icons and small pictures

A set of Windows 10 style icons are provided for selection under the "icons" list and the "small pictures" list in the Property tab for controls. The developer can quickly find out

these new Windows 10 icons, according to the text appended to the icon name, for example, "_icon_2" is appended to the icon name, and "_2" is appended to the small picture name.

1.6 64-bit enhancements

PowerBuilder 2019 has enhanced support for 64-bit, so the following bugs no longer exist:

1. Previously, when developers used the registry functions (including RegistryDelete, RegistryKeys, RegistryGet, RegistrySet, & RegistryValues) to access the 64-bit registry entries in a 64-bit operating system, they were incorrectly redirected to the Wow6432Node registry entry; and now they can access the correct registry entry.

   And to support this, a longlong type enumeration value (RegLongLong!) is added for the RegistryGet and RegistrySet functions. For more information, see Section 10.509, “RegistryGet” in *PowerScript Reference* or Section 10.511, “RegistrySet” in *PowerScript Reference*.

2. Previously, 32-bit PowerBuilder application and 64-bit PowerBuilder application cannot coexist on the same client machine because they used the same location to store runtime files; and now they use different locations, so they can co-exist on the same machine.

1.7 PBC enhancements

PBC (PowerBuilder Compiler) supports a new parameter "/pd" which can specify whether to generate the PBD/DLL file for a PBL.

1.8 New online installer -- Appeon Installer

Starting from 2019, a new and more efficient way to install Appeon products is introduced -- Appeon Installer, which is an online installer with a self-extracting download that leads you through the installation process. The machine will be required to connect with Internet during the installation process.

For detailed instructions on using Appeon Installer, refer to Section 3.1, “Online Installation” in *Installation Guide*.

1.9 Classified features

While new features are added in PowerBuilder, some of the existing features might no longer be needed or recommended for use. You will find the following three classifications of features in the PowerBuilder Help.

- **Discontinued** - feature that has been completely removed from the product.

  For example, EAServer projects/targets, PowerBuilder .NET IDE, and Windows Form projects/targets are discontinued features.

- **Obsolete** - feature that is available, but is no longer eligible for technical support and will no longer be enhanced.

  For example, Web DataWindow, SOAP client, Web service DataWindow (OData and SOAP), .NET Web service target, .NET assembly target are obsolete features.

- **Stable** – feature that is available and is still eligible for technical support, but will no longer be enhanced.

2 Bug Fixes & Known Issues in PowerBuilder 2019

The bug fixes and known issues for PowerBuilder 2019 are listed in the Release Bulletin for PowerBuilder at the following links:

Bug Fixes for PowerBuilder: https://www.appeon.com/support/documents/appeon_online_help/pb2019/release_bulletin_for_pb/bug_fixes.html

Known Issues for PowerBuilder: https://www.appeon.com/support/documents/appeon_online_help/pb2019/release_bulletin_for_pb/known_issues.html