Desktop Utility

Versions:

Version ID	Date	Changes
1.0	11/04/2020	Initialized the document with the Utility architecture

Desktop Utility Architecture

For our project, we needed to create a Desktop application, that could take a bulk of IFC files and convert them into the 3D models. These 3D models then will be used by the Web Applications where the users can interact with them. Not every product will have the 3D model with them, but most of the Holyoake products are going to have them.

In addition to the IFC file conversion, the desktop utility also convert the textfiles that contain the parameterized information for these 3D models. This information is extracted from the Textfiles and then converted in to a JSON format. The data in JSON format is sent to the backend for storage. This parameterized data is also shown to the user at the frontend. Once again, the parameterized data is only for the products that have 3D models.

The desktop utility was created by our team during this project and it was created from scratch. The utility was coded in C# using .Net Framework 4.7. The utility is made using the WPF (Windows Presentation Forms) technology and makes use of MVVM (Model-View-ViewModel) architecture for allow for a decoupled code.

In addition to the MVVM design pattern, a Service Layer pattern is also applied. All the processing like Textfile processing, or IFC processing is done in this service layer. This was once again done to provide separation of concern. In the long run, this will make code easy to extend.

The application makes use of third party libraries or nuget packages to get the desired results. However, these packages are restored automatically when the project is downloaded and built again. The important folders in the project structure are discussed below.

The file structure of the project can be seen below. Details of the different folders is provided at the end of the document.

Directory Structure

```
+---IFCConverto
|   +---Convertors
|   +---Enums
|   +---Helpers
|   +---Models
|   +---MVVM
|   +---Properties
|   +---Services
|   +---Themes
|   +---ViewModels
|   \---Views
\---packages
    +---AWSSDK.Core.3.5.1.26
    |   +---lib
    |   |   +---net35
    |   |   +---net45
    |   |   +---netcoreapp3.1
    |   |   +---netstandard1.3
    |   |   \---netstandard2.0
    |   \---tools
    +---AWSSDK.S3.3.5.3.4
    |   +---analyzers
    |   |   \---dotnet
    |   |       \---cs
    |   +---lib
    |   |   +---net35
    |   |   +---net45
    |   |   +---netcoreapp3.1
    |   |   +---netstandard1.3
    |   |   \---netstandard2.0
    |   \---tools
    +---ControlzEx.4.3.2
    |   \---lib
    |       +---net45
    |       +---net462
    |       +---netcoreapp3.0
```

```
|       \---netcoreapp3.1
+---MahApps.Metro.2.2.0
|   \---lib
|       +---net452
|       +---net46
|       +---net47
|       +---netcoreapp3.0
|       \---netcoreapp3.1
+---MahApps.Metro.IconPacks.4.5.0
|   \---lib
|       +---net45
|       +---net46
|       +---net47
|       +---netcoreapp3.0
|       +---netcoreapp3.1
|       \---uap10.0.16299
|           +---MahApps.Metro.IconPacks
|           +---MahApps.Metro.IconPacks.BoxIcons
|           |   \---Themes
|           +---MahApps.Metro.IconPacks.Entypo
|           |   \---Themes
|           +---MahApps.Metro.IconPacks.EvaIcons
|           |   \---Themes
|           +---MahApps.Metro.IconPacks.FeatherIcons
|           |   \---Themes
|           +---MahApps.Metro.IconPacks.FontAwesome
|           |   \---Themes
|           +---MahApps.Metro.IconPacks.Ionicons
|           |   \---Themes
|           +---MahApps.Metro.IconPacks.JamIcons
|           |   \---Themes
|           +---MahApps.Metro.IconPacks.Material
|           |   \---Themes
|           +---MahApps.Metro.IconPacks.MaterialDesign
|           |   \---Themes
|           +---MahApps.Metro.IconPacks.MaterialLight
|           |   \---Themes
|           +---MahApps.Metro.IconPacks.Microns
|           |   \---Themes
|           +---MahApps.Metro.IconPacks.Modern
|           |   \---Themes
|           +---MahApps.Metro.IconPacks.Octicons
|           |   \---Themes
|           +---MahApps.Metro.IconPacks.PicolIcons
|           |   \---Themes
|           +---MahApps.Metro.IconPacks.RPGAwesome
|           |   \---Themes
|           +---MahApps.Metro.IconPacks.SimpleIcons
|           |   \---Themes
|           +---MahApps.Metro.IconPacks.Typicons
|           |   \---Themes
|           +---MahApps.Metro.IconPacks.Unicons
|           |   \---Themes
|           +---MahApps.Metro.IconPacks.WeatherIcons
|           |   \---Themes
|           \---MahApps.Metro.IconPacks.Zondicons
|               \---Themes
+---Microsoft.Xaml.Behaviors.Wpf.1.1.19
|   +---lib
|   |   +---net45
|   |   |   \---Design
|   |   \---netcoreapp3.0
|   |       \---Design
|   \---tools
+---Newtonsoft.Json.12.0.3
|   \---lib
|       +---net20
|       +---net35
|       +---net40
|       +---net45
|       +---netstandard1.0
```

```
|       +---netstandard1.3
|       +---netstandard2.0
|       +---portable-net40+sl5+win8+wp8+wpa81
|       \---portable-net45+win8+wp8+wpa81
+---WindowsAPICodePack-Core.1.1.1
|   \---lib
\---WindowsAPICodePack-Shell.1.1.1
    \---lib
```

Convertor Folder

It contains all the convertor that are consumed by the UI / XAML layer.

Enums

This folder contains all the Enums that are used across the entire application

Helpers

This folder contain the Helper classes which are consumed by service layer to perform certain actions

Models

These are the Model DTO (Data Transfer Object) classes. These models are converted in to the JSON and then passed on to the Backend for further processing.

MVVM

This folder contains the basics classes that are required for MVVM pattern. The solution doesn't use third party MVVM library, instead a custom basic level MVVM architecture is coded for this application

Services

All the services that are consumed by the viewmodels to get the job done are present here. This folder also contain services that are responsible for uploading the data to the AWS and sending it to the Backend

Themes

All the styling of the controls are done in separate theme files. There are no inline styling done in the application. This folder contain the theme files for different controls.

ViewModels

This folder contains all the viewmodels that are binded with the view layer. (Think of viewmodels as the controller layer and the view as the presentation layer to understand the concept clearly)

Views

This is the presentation layer for the Desktop Utility. All the views that are shown to the user are placed in this folder. The view layer makes use of viewmodel layer to serve the required content to the user