Revitron UI

Marc Anton Dahmen

Nov 29, 2022

REVITRON UI

Revitron UI is a pyRevit extension written in Python based on Revitron. It helps you to work efficiently with complex models by automating common tasks.

Docs GitHub

CHAPTER

ONE

GETTING STARTED

The Revitron UI extension requires the Revitron library to be installed as well as a pyRevit extension. It is possible to install both extensions manually or as a bundle together with a fork of pyRevit. Generally it is recommendend to install the bundled version as described below.

Note: The bundle installer uses Git to manage dependencies. Please make sure that Git is installed properly on your system before installing Revitron.

1.1 Bundle Installer

To install the full bundle including pyRevit, Revitron and the Revitron UI, follow the instructions below:

1. In case Git is not already installed — download and install Git.
2. Right-click here to download the installer. Make sure it keeps the `.bat` extension.
3. Move the `install.bat` to the directory, where you want to install pyRevit.
4. Double-click the `install.bat` file.
5. Start **Revit**.

1.2 Manual Installation

The single library and UI packages can be installed using the pyRevit CLI as follows:

```
pyrevit extend lib revitron https://github.com/revitron/revitron.git
pyrevit extend ui revitron https://github.com/revitron/revitron-ui.git
```

Alternatively the packages can also just be cloned with Git as follows:

```
cd C:[\path\to\pyrevit]\extensions
git clone https://github.com/revitron/revitron.git revitron.lib
git clone https://github.com/revitron/revitron-ui.git revitron-ui.extension
```

CHAPTER

TWO

USER GUIDE

2.1 Export

2.1.1 Export Sheets as PDF

2.1.2 Export Sheets as DWG

2.1.3 Export Settings

The export settings control both — the PDF and the DWG export. Before being able to export sheets from Revit, please make sure to configure the main settings correctly as described below.

Sheet Export Directory The final output root directory for the exported and correctly named PDFs.

Sheet Naming Template (optional) The template for the exported files. Parameter names can be wrapped in `{}` and will be substituted with their value on export like `{Sector}\{Sheet Number}-{Sheet Name}`. Note that the template should not have any extension. Defaults to `{Sheet Number}-{Sheet Name}`.

Sheet Size Parameter Name The name of the paramter of the sheet category where a string to define a paper size can be stored. The values should match the paper sizes defined in the PDF printer's configuration like for example `A4` or `A0`.

Default Sheet Size The default paper size for all sheets where no value is set for the sheet size paramter.

Sheet Orientation Parameter Name The name of the paramter of the sheet category where a string to define a the sheet orientation can be stored.

Default Sheet Orientation (optional) The default sheet orientation for all sheets where no value is set for the sheet orientation paramter. Defaults to `Landscape`.

PDF Settings

To be able to export PDFs, there are two more required settings to be configured. Note that the PDF exporter expects a network PDF printer to be running.

PDF Printer Address The network address of the printer like `\\server\printer`.

PDF Temporary Output Path The path of the output directory of the network PDF printer. The script will take the incoming PDFs from that location and move it **Sheet Export Directory** (see below).

DWG Settings

In order to export DWG, the export options for the project have to be defined in the settings.

DWG Export Setup The name of the export setup configured in the Revit settings.

2.2 History

2.3 Links

2.3.1 Move Links

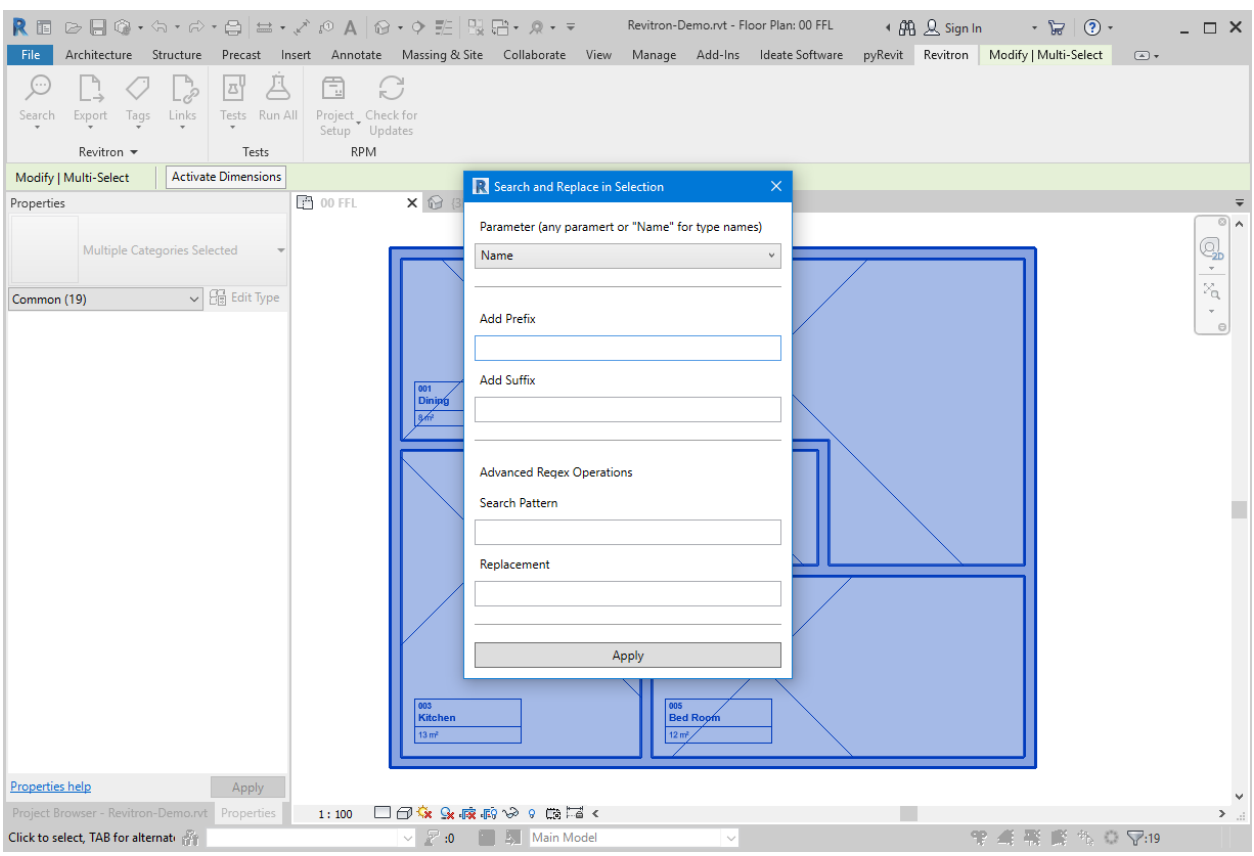
2.3.2 List Link Paths

2.3.3 Search and Replace in Link Paths

2.4 Model

2.5 Replace

The **Replace in Selection** tool let's you perform a *search and replace* action on a selected parameter for all elements in the current selection. It is also p



2.6 Rooms

2.7 Packages

Since the **Revitron** API enables you to easily create specific extensions for certain projects, you also want to be able to easily roll out those extension to your team without individually installing them manually. The **Revitron Package Manager** (RPM)- takes care of that.

2.7.1 Installing Dependencies

The package manager basically lets you define a list of **pyRevit** extensions and stores it in Revitron's Document Config Storage. Since your list of dependencies becomes then part of your Revit model you can just synchronize your local file to distribute it to other team members. To actually load the extension tools, you can hit the **Install Extension** button when needed at any time.

2.7.2 Updating Packages

The package manager can also search for available updates automatically. When starting Revit, the updater will check for available updates of both — the pyRevit core and any installed extension.

Note: The updater shipping with the Revitron UI requires Git to be installed properly on your System in order to pull changes and check for updates.

In case there are pending updates waiting to be installed they can just be applied with a single click. In contrast to extension updates, core updates require a shutdown of **all** running Revit instances to proceed. So make sure, all open Revit files are saved before installing core updates! Note that it is also possible to check for pending updates manually at any time hitting the **Check for Updates** button in the RPM panel.

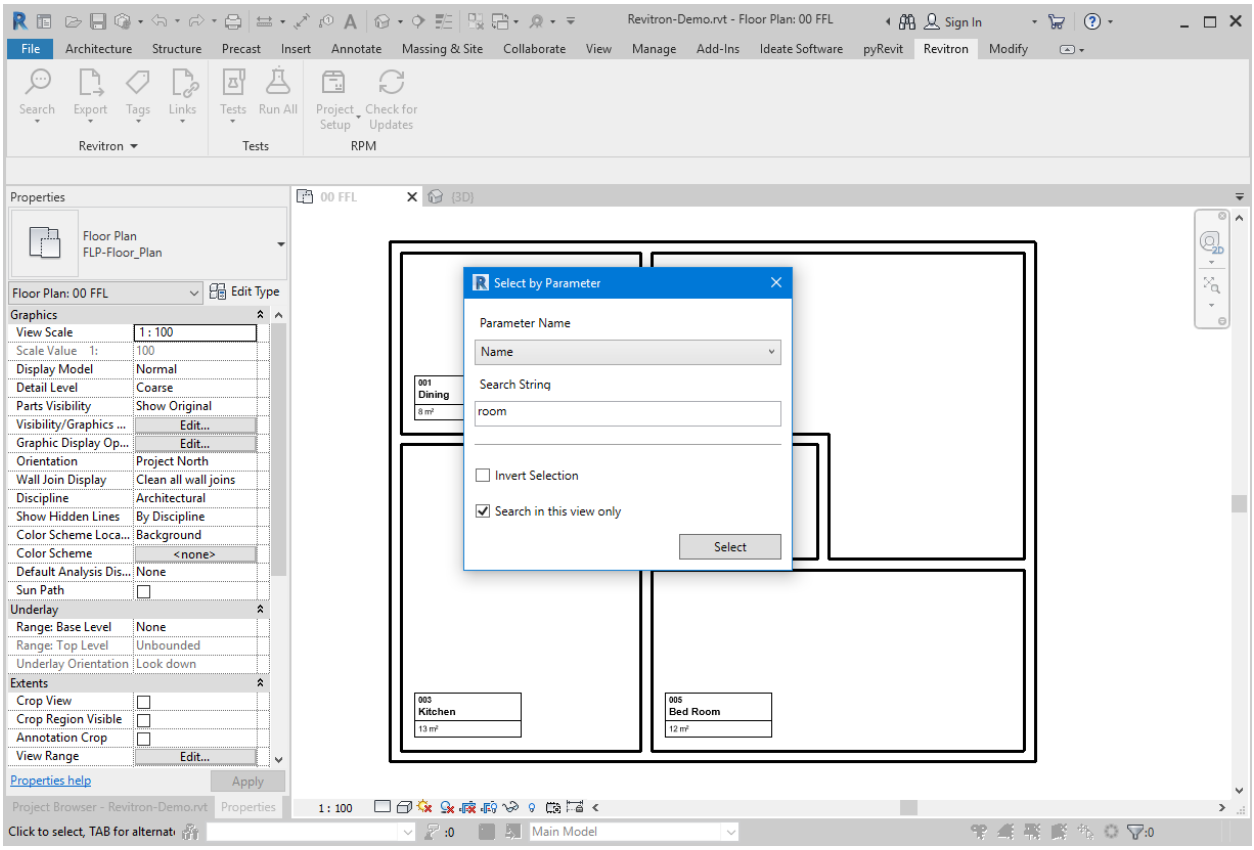
Forcing Updates

In some cases you may end up with a *dirty* working copy in one of the installed extensions and the updater will not stop asking you to update on every Revit start, because it won't update *dirty* repositories. To easily discard all local changes to the extension repositories, you can force updates by hitting **Force Extension Updates** in the dropdown below the **Check for Updates** button. Note that for now this is only possible for extension repositories and won't work with the pyRevit core.

2.8 Select

The Search pulldown provides tools to search for elements by parameter values and to batch modify parameter values within a set of selected elements.

2.8.1 Select Elements by String



2.9 Tags

The **Tags** pulldown contains tools to make tagging more convenient. The collection of tools is still work in progress and will grow over time.

2.9.1 Rooms

The following five buttons are available for tagging all rooms in the current selection. If nothing is selected, all rooms in the active view will be tagged.

- Tag Rooms > Center
- Tag Rooms > Top Left
- Tag Rooms > Top Right
- Tag Rooms > Bottom Left
- Tag Rooms > Bottom Right

2.10 Utils