http://excel.fit.vutbr.cz

Complex Home Automation in Home Assistant using NetDaemon

Pavel Yadlouski

Abstract

Home automation is no longer a dark magic. You can do it yourself. Quickly, you would realize that you need to create your own automation scenarios. After some research and experiments, you will most likely end up with a huge amount of repetitive code and configuration. This paper introduces a new approach to creating and sharing home automation scenarios in Home Assistant.
The core idea is to create a library (NetEntityAutomation) that will provide out-of-the-box plug-&-play automation with only one user requirement: to provide devices to work with.
The result of this work is already used in the authors' home on daily bases and shared with the Home Assistant community. While having several bugs related to business logic of the automation, the idea is proved to be working.
NetEntityAutomation library opens an alternative way for using, creating, and sharing automation scenarios using intuitive flows and classic technologies.

*xyadlo00@vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

There is nothing new about having home automation in your home. Today's question is not if you need to have any home automation, but how to create one in a better way and how to share your scenario afterwards. This project aims to provide an alternative way for complex home automation with a declarative programming approach. **Home Assistant (HA)** is chosen as the base platform for home automation.

Currently, the main HA approach to creating and sharing automation scenarios has several weaknesses. Such automation is created from the UI (mostly by beginners) or written manually in the YAML format. The amount of code that must be written to create simple automation is enormous [Figure 1]. In addition, it will be duplicated for any other room with the same automation. Sharing automation in such a format is extremely difficult. The stability and reliability of such automations is another bottleneck.

Home Assistant has a native solution to this problem. In HA you can create the so-called *Blueprint*[1] and share it on the platform to share such automations specifically for HA. This approach is extremely (beginner) user-friendly. However, Blueprints still cannot provide any out-of-the-box solution for providing expected result in any case and still require code duplication.

The core idea of the solution is to use the NetDaemon framework to work with abstractions over real entities in C# language. Having a strongly typed programming language environment, there are limitless opportunities to build automation flows. In this case, the key point is to provide a **reliable** library for building and managing automations. The reliability means an expected state even in corner cases, such as system reboot, entity becomes available, etc. Such a requirement is fulfilled with finite state machine (FSM) that stores its state in persistent storage with predefined steps what to do in corner cases.

Currently, NetEntityAutomation is used for author's home automation. Having an FSM that represents the state of a single entity with storing the state to the file system proves to be a robust solution. The implemented architecture along with the technologies used provides a wide range of extensions. One of them is to visualize and control the state of automation in runtime. Currently, the visualization is made in a standard HA way through the provided sensor entity that represents the state of individual automation. The control is made by exposing related services to HA.

2. Details

This project is made for the Home Assistant environment as it provides extensive API and freedom for development of any kind. Other popular platforms, such as Apple HomeKit and Google Home, are also considered, but were rejected in the functionality analysis step. As an application framework, NetDaemon is used. NetDaemon [2] is a C#-based framework that first aimed to provide an alternative to AppDaemon (a Python-based framework for **simple** automation tasks [3]). The created library is called NetEntityAutomation (NEA).

2.1 HA Approach VS NEA Approach

The default approach in Home Assistant to create automation through the UI that creates the YAML file is a good approach [Figure 1]. However, it has limitations, that will popup very quickly:

- No automation state management in corner case (system restart, entity unavailability, etc.)
- No grouping of the automation to related rooms, what leads to hard managements
- Configuration duplication
- Somehow hard to read in raw format

In NEA, an intuitive concept of the room [Figure 2] is used to group the automations. The room is something that contains several entities that can be automated in some popular way. For example, motion-triggered light or sun-position-triggered blinds. Each of these automations is a common scenario for many home-owners. NEA can be seen as the provider of popular Blueprints, but implemented in C# with additional features that are easily implemented in the programming language and hard to imagine in the HA blueprint. In simple words, in NEA instead having user to create the entire automation, there is a declarative way to define the entities for chosen automation. The rest of configuration can be done by NEA.

2.2 Implementation

The design of NetEntityAutomation is based on the changes made in NetDaemon, which provides **generic interfaces** for HA entities [Figure 3]. These interfaces act as a bridge between external code, such as NEA, and entities that are generated to C# classes from HA using NetDaemon. In NEA, **generic interfaces are used to implement automation scenarios**.

The state of each entity, related to a given automation, **is represented in FSM** [Figure 4] FSM also defines the transition of allowed state for a given entity. After each successful FSM transition,**the current state is stored in the file system in the JSON file**. This file is used to set the correct state of the entity. Given automation provides the logic for setting up the correct state based on current conditions.

2.3 Deployment

Home Assistant itself provides several deployment options: as a specific OS, on a virtual machine, and in a container. As NEA runs as a NetDaemon application, it uses the environment provided by NetDaemon. In this project, the containerized option of the NetDaemon deployment is used. This approach makes the NetDaemon an **independent service** that can be moved to any platform that supports containerization. Communication with HA is made through the REST API provided by HA.

2.4 Automation sharing

Each automation can be implemented in separate C# project with dependency on the NEA core package. The sharing of conditioned automation is natively supported. Such projects are **packed into a NuGet package** [4] and shared on any preferred platform. Using custom automation is as easy as installing the corresponding NuGet package and configure with NetDaemon-generated entities. The rest is left to NetEntityAutomation.

3. Conclusions

This work deals with an alternative approach to creating and sharing home automation scenarios. The core ideas of this project came from real-world experience in developing any other open-source software and sharing it. I am really proud that this work has received several attention from NetDaemon developers and has big potential based on the feedback.

Acknowledgements

I would like to thank my supervisor **Ing. Jan Pluskal Ph.D.** and **NetDaemon developers** for their help!

References

[1] About blueprints. https://www.home-assistant.io/docs/blueprint/.

[2] Netdaemon. https://netdaemon.xyz/docs/user/, url = https://netdaemon.xyz/docs/user/,.

[3] Welcome to appdaemon's documentation! https://appdaemon.readthedocs.io/en/latest/, url = https://appdaemon.readthedocs.io/en/latest/,.

[4] Netentityautomation-custom-automatation. https://github.com/x00Pavel/NetEntityAutomation-Custom-Automatation, url = https://github.com/x00Pavel/NetEntityAutomation-Custom-Automatation,.