Build DNNrocket Event Plugin

# Introduction

We can create event plugins in DNNrocket. These event providers are triggered by the API call. This document explains how to build an event plugin.

# Start

Create Visual Studio project in the /DesktopModules/DNNrocketPlugins. With a “.Net Standard” class library. (Folder name can be any company folder under desktopmodules.)

Rename class library to “Events.cs”, just to give a standard name across modules. (CamelCase)

set a reference to “DNNrocket API” and “Simplisity”.

The Events class should inherit from “DNNrocketAPI.EventInterface”. And implement the abstract methods.

In after build copy the modules:

copy "$(ProjectDir)$(OutDir)$(TargetFileName)" "$(ProjectDir)..\..\..\bin\$(TargetFileName)"

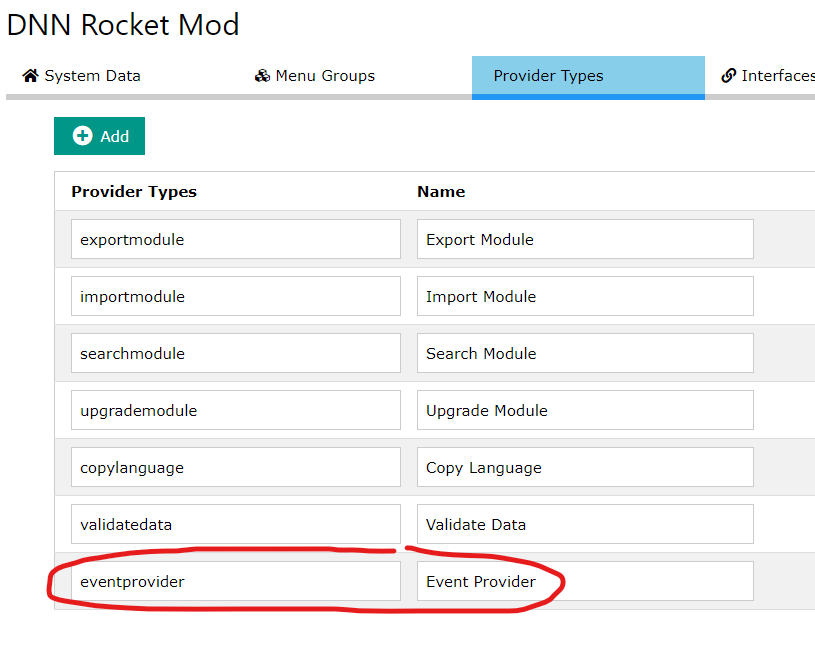
copy "$(ProjectDir)$(OutDir)$(AssemblyName).pdb" "$(ProjectDir)..\..\..\bin\$(AssemblyName).pdb"

The event methods can now be created.

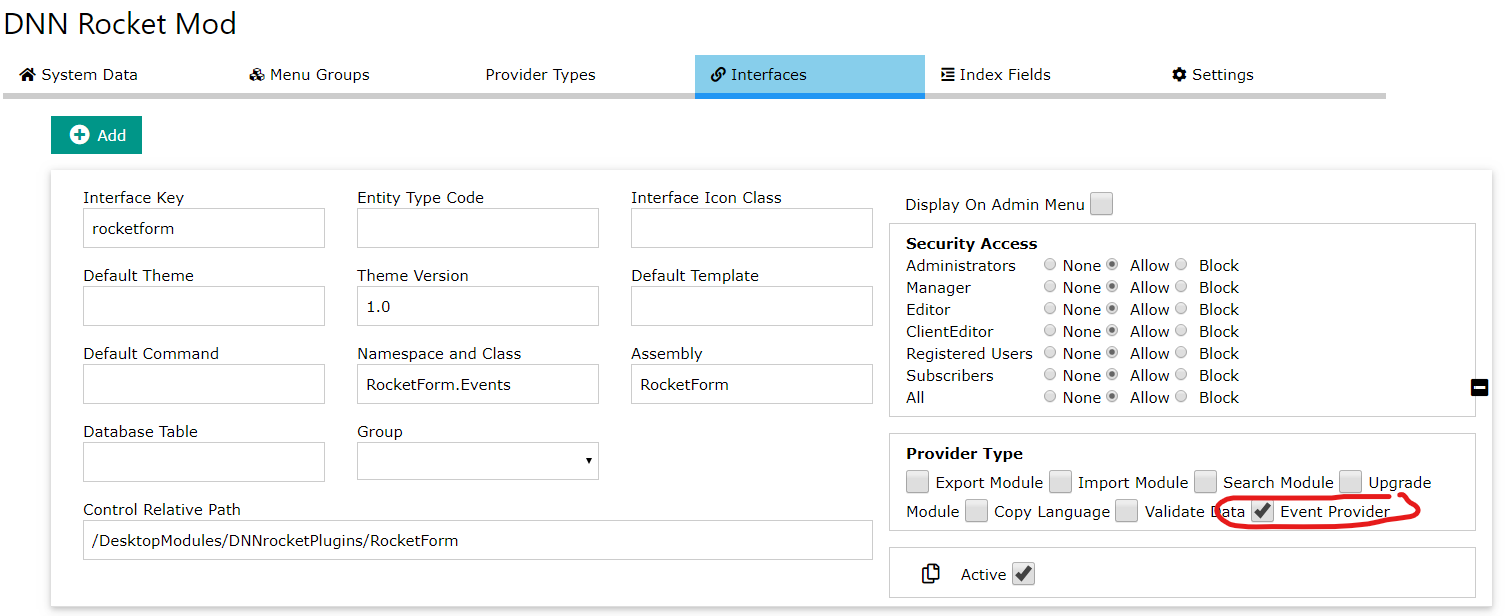
# Registering the Events

The DNNrocket event interface has 2 methods, “BeforeEvent” and “AfterEvent”. These events are called by the API before and after running an API command. To be triggered we need to register the event provider.

For an event provider we must has an event provider defined in the Rocket System as a “eventprovider” provider. [/DesktopModules/DNNrocket/adminsystem.html]



In the DNNrocket system that the event is linked to, create an extra interface which identifies the assembly, class and event provider.



Now we only need to move the assembly into the DNN bin folder for the event to be triggered.

# How it Works

The event providers are called by the DNNrocket API, specifically in the “DNNrocketAPI.ApiControllers.RocketController” class. There are a number of actions, like login and side menu call that will be performed before the event call. The “before event” call is called just prior to the system API action and the “after event” call is made just after the API action.

It looks for the event provider in the system interfaces and calls each provider defined. The API will call the event provider blindly, so any security required MUST be setup in the event code.

The code looks for a return from the event, the event return is data based on “paramData” and “postData”. The concept is that if this data needs to change, before or after, the API action we can alter the data in the event and then the new data can be used.

// before event

var rtnDictInfo = DNNrocketUtils.EventProviderBefore(paramCmd, systemData, postInfo, paramInfo, \_editlang);

if (rtnDictInfo.ContainsKey("post")) postInfo = rtnDictInfo["post"];

if (rtnDictInfo.ContainsKey("param")) paramInfo = rtnDictInfo["param"];

---------- API CALL ----------

// after Event

returnDictionary = DNNrocketUtils.EventProviderAfter(paramCmd, systemData, postInfo, paramInfo, \_editlang);

if (returnDictionary.ContainsKey("outputhtml")) strOut = returnDictionary["outputhtml"];

if (returnDictionary.ContainsKey("outputjson")) strJson = returnDictionary["outputjson"];

# Changing the return data

If we DO NOT want to return any data, we can simply return a blank dictionary.

public override Dictionary<string, SimplisityInfo> BeforeEvent(string paramCmd, SystemData systemData, SimplisityInfo interfaceInfo, SimplisityInfo postInfo, SimplisityInfo paramInfo, string langRequired = "")

{

//Before event

var rtnData = new Dictionary<string, SimplisityInfo>();

return rtnData;

}

If we want to change data then we can return the required values in the return dictionary, but we MUST use the correct key for the data in the dictionary.

public override Dictionary<string, string> AfterEvent(string paramCmd, SystemData systemData, SimplisityInfo interfaceInfo, SimplisityInfo postInfo, SimplisityInfo paramInfo, string langRequired = "")

{

//After Event

var rtnData = new Dictionary<string, string>();

rtnData.Add("outputhtml", "<h1>TEST EVENT</h1>");

return rtnData;

}

# BEFORE event return Dictionary keys

|  |  |  |
| --- | --- | --- |
| Key | Data Type | Help |
| post | SimplisityInfo | This class is returned to the “postInfo” variable that was past. |
| param | SimplisityInfo | This class is returned to the “paramInfo” variable that was past. |

# AFTER event return Dictionary keys

|  |  |  |
| --- | --- | --- |
| Key | Data Type | Help |
| outputhtml | String | The html output that will be returned to the client. |
| outputjson | String | The json output that will be returned to the client. |