relevant ecommerce™

TECHNICAL

ZNODE PLUGIN DEVELOPER GUIDE

September 2015

Table of Contents

Extending Znode

Znode is extended by building plugins. Znode plugins are classified as Admin, API, and BackgroundTask aligning with the Znode process hosting the plugin.

Admin plugins are hosted in the Znode.Engine.MvcAdmin website. These plugins extend the Znode administration console allowing Znode administrators to configure a plugin.

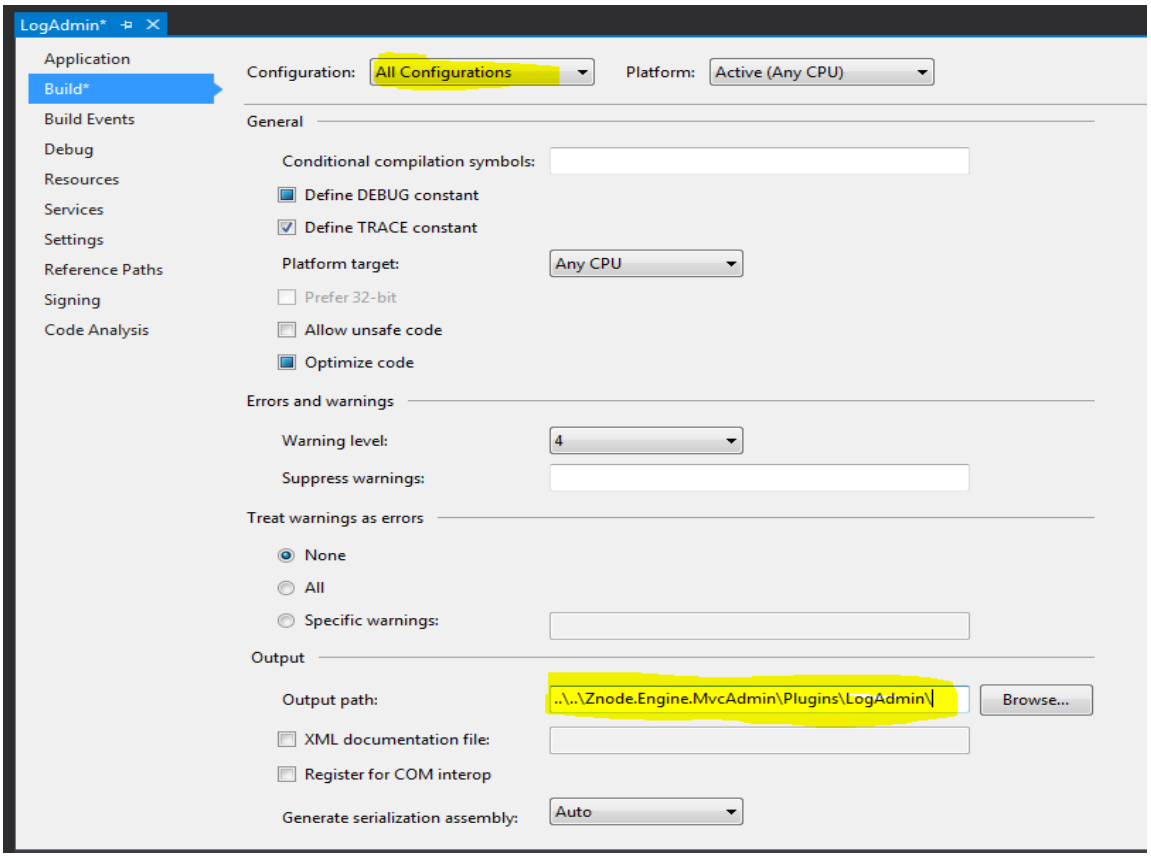
API plugins are hosted in Znode.Engine.API website. These plugins extend the Znode rest API typically to expose data in the Znode platform.

Admin Plugin Development

Admin plugins provide a user interface for Znode administrators to configure a plugin. Admin plugins are a Visual Studio Class Library project configured to build into the Znode.Engine.MvcAdmin\Plugins folder. On application start, Znode.Engine.MvcAdmin reviews plugins in this folder and attempts to load each plugin.

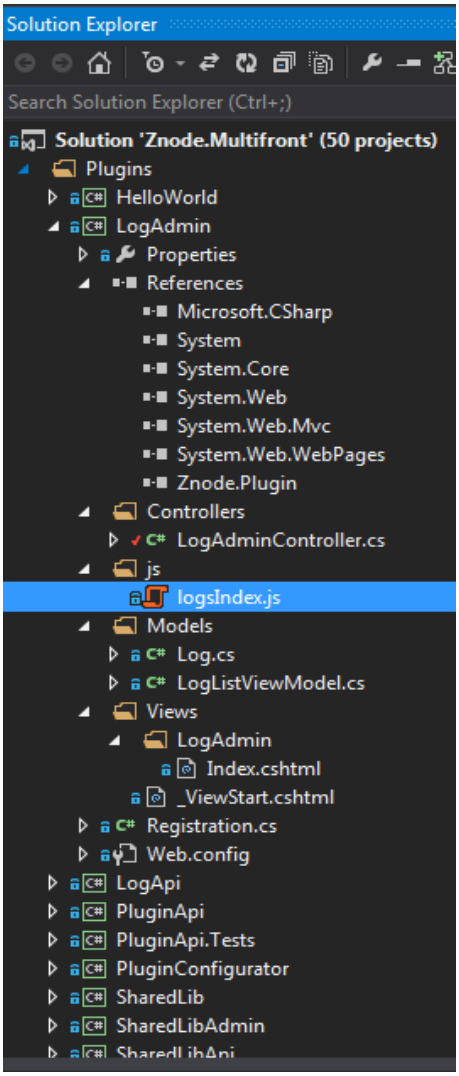
Project Settings

Admin plugin project properties need to be customized to put the build product in a location where Znode.Engine.MvcAdmin is configured to look for plugins. Below you can see that for All Configurations, the Output Path has been configured to copy build output to a relative path to the Znode.Engine.MvcAdmin plugins folder ..\..\Znode.Engine.MvcAdmin\Plugins\LogAdmin\ where LogAdmin is a folder for this specific plugin. You will change LogAdmin part of this path to be specific to your plugin.



Project Structure

This section discusses the project structure by discussing the LogAdmin example admin plugin in the screenshot below. The fastest way to get started creating a plugin is to copy an example shipped with the product and rename folders and files to an appropriate name. Resharper assists in speeding up this process.



The LogAdmin project is the example project being discussed in this example. Notice the LogAdmin project follows ASP.NET MVC folder and file naming conventions. Since this is a class library project, ASP.NET MVC scaffolding is not available to assist the developer with maintaining these conventions.

References

The LogAdmin plugin references System.* libraries from Microsoft, and these references **MUST** be referenced from the same location as the host process. For example, System.Web.Mvc must be referenced from *SharedLibraries\NuGet_Packages\Microsoft.AspNet.Mvc.5.2.3\lib\net45\System.Web.Mvc.dll*. Do NOT use NuGet to add references to your admin plugin when your admin requires a reference also reference from the hosting process (*Znode.Engine.MvcAdmin*). By default, NuGet pulls the most recent version of assemblies and will lead to version conflicts when your plugin attempts to load into *Znode.Engine.MvcAdmin* website.

Controllers

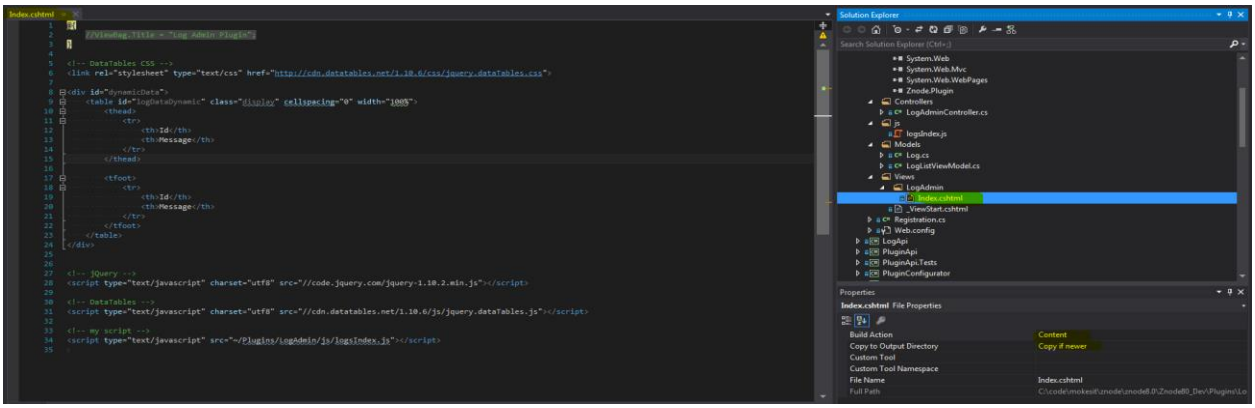
Admin plugin controllers are normal MVC controllers except for the way admin plugin reference their views.

```
Index.cshtml    logsIndex.js    LogAdminController.cs
LogAdmin                        LogAdmin.Controllers.LogAdminController
1   using System.Web.Mvc;
2
3   namespace LogAdmin.Controllers
4   {
        1 reference | rmokros@mokesit.com, 29 days ago | 1 change
5       public class LogAdminController : Controller
6       {
7
            0 references | rmokros@mokesit.com, 29 days ago | 1 change
8           public LogAdminController()
9           {
10
11
12
13          }
14
            0 references | rmokros@mokesit.com, 29 days ago | 1 change
15          public ActionResult Index()
16          {
17              return View("~/Plugins/LogAdmin/Views/LogAdmin/Index.cshtml");
18          }
19      }
20  }
21
```

Views must be referenced using a virtual path location relative to where the plugin will be installed in the Znode.Engine.MvcAdmin website.

Views

Views are default ASP.NET MVC Razor views with one caveat that on the properties folder, **Build Action** and **Copy to Output Directory** MUST be set to copy the view as part of the build output. This makes sure the view gets copied as part of the build process into the plugins folder.



Common Layout

_ViewStart.cshtml specifies the default layout as *~/Views/Shared/_Layout.cshtml* resulting in a consistent layout for Znode plugins. This is not a requirement, and plugin developers are free to customize the layout. As with views, set **Build Action** to *Content* and **Copy To Output Directory** to *true* to ensure this file is copied as part of the build output.

Registration.cs

Registration.cs implements Znode.Plugin.IAdminPlugin interface. This implementation can be copied with the plugin developer changing the Title and EntryControllerName to appropriate values for your plugin.

- Use IAdminPlugin.Install to install database schema associated with your plugin.
- Use IAdminPlugin.Uninstall to uninstall database schema associated with your plugin.

Web.Config

The web.config file is **required** and this file shares **Build Action** and **Copy to Output Directory** settings from Views and _ViewStart.cshtml of *Content* and *Copy if newer*, respectively.

Javascript Resources

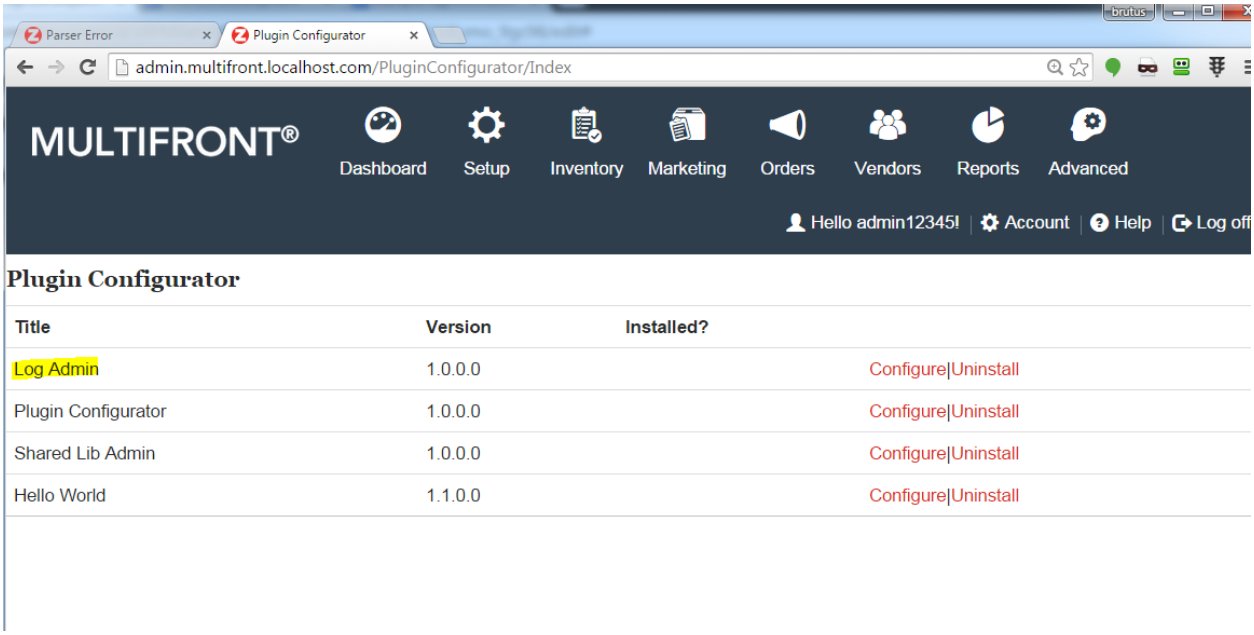
Admin plugins can include javascript files. Javascript files must be copied to the output directory, so each must be marked with File Properties **Build Action** set to *Content* and **Copy to Output Directory** set to *Copy if newer*.

In an admin plugin, reference plugin javascript files using the virtual location with respect to the plugin build location. In the case of the LogAdmin example, the Index.cshtml references its javascript file using the following razor syntax.

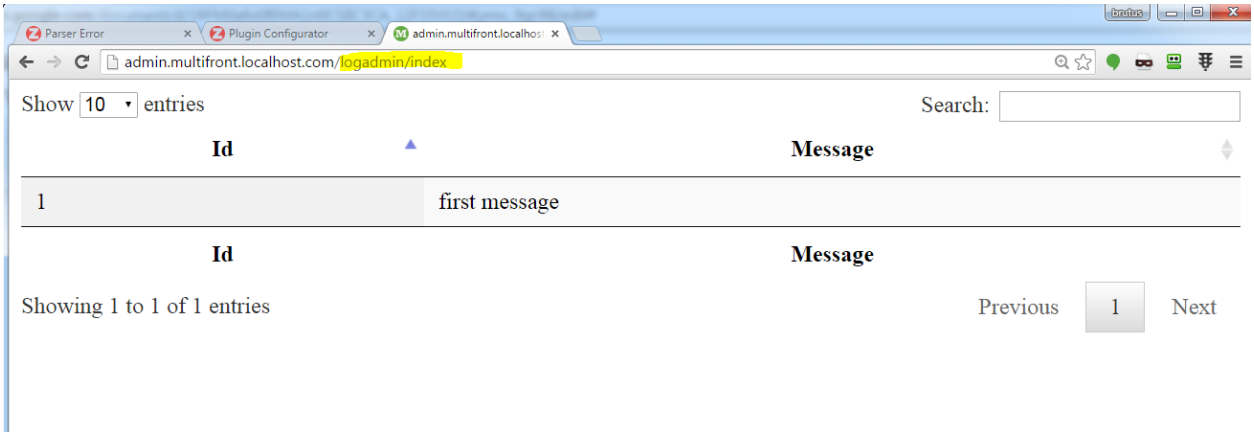
```
<script type="text/javascript" src="~/Plugins/LogAdmin/js/logsIndex.js"></script>
```

Viewing Admin Plugin

To view your admin plugin, log into Multifront Admin and select ***Setup*** -> ***Plugin Configurator*** from the menu.



Navigate to your plugin using mvc routes hosted by your plugin. For LogAdmin, the route is *LogAdmin/Index*.



Admin Plugin Developer Tricks

This section identifies some common issues and resolution.

Views not updating

When hosting Znode.Engine.MvcAdmin in IIS on a development workstation, you are tempted to change and save views, but after refreshing your browser, your changes are not visible. The problem is your admin plugin only copies your views to the build output folder after building your admin plugin. So, although this technique works on normal ASP.NET MVC projects, it will not work in your Znode admin plugin because your views must be copied to the Znode.Engine.MvcAdmin plugins folder before Znode.Engine.MvcAdmin will recognize your changes. Depending on your changes, an IISReset may also be required to fire the Application_Start event where Znode.Engine.MvcAdmin will reload your plugin.

Data Access

Znode performs all data access through the rest API hosted by the Znode project Znode.Engine.API. If you admin plugin requires data access, build a Znode API plugin to expose your data.

Plugins Menu

Menu for plugins can be added by making entry in database tables or you can use the below code if you want to add a new Tab in already existing Views of Admin.

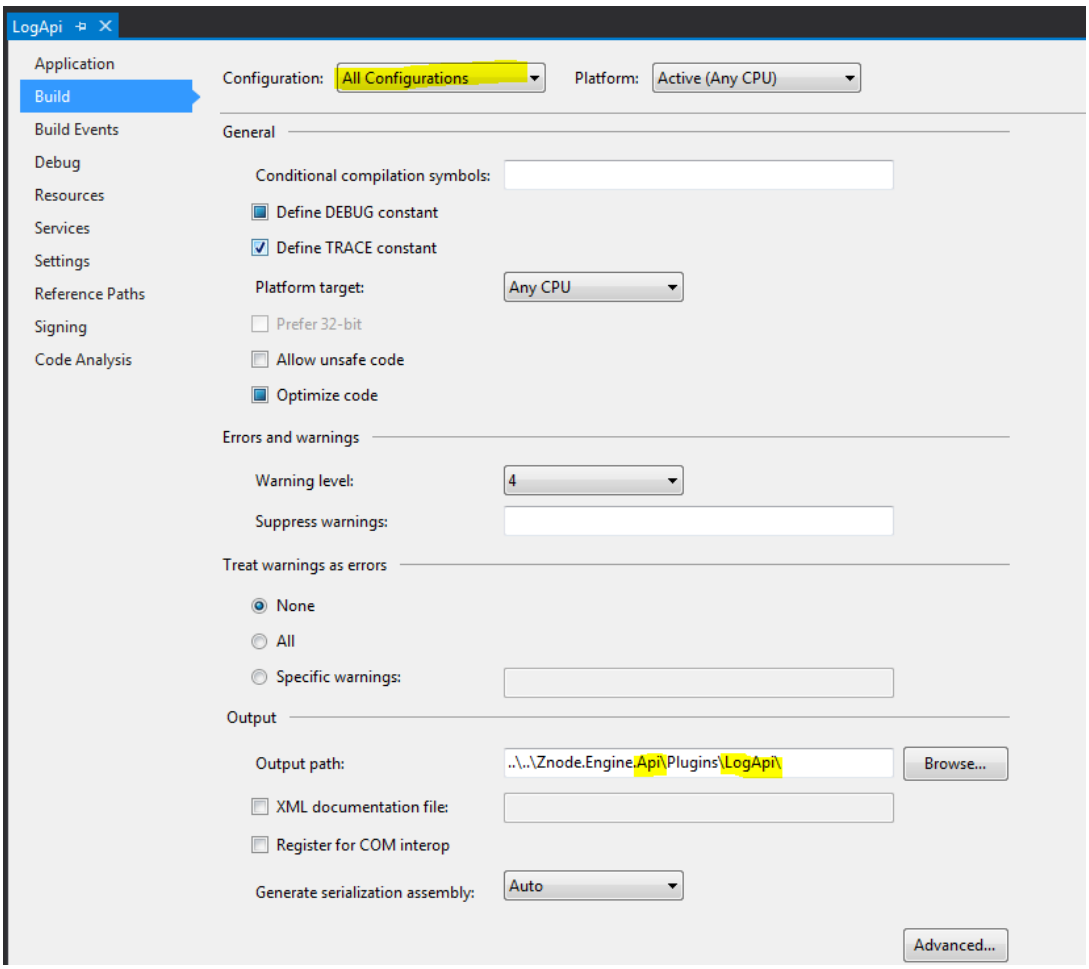
```
<script>
   $(function () {
      $('#tabManageProduct').append('<li><a href=""/RecurringOrderAdmin/Index"" id=""recurringorderlist"" data-queryparam=""id=@Model.CustomerAccount.AccountId"" data-targetitemid=""panel1"" data-isignorgrid=""true"" onclick=""CommonHelper.Tab_Change_Handler(this);"" data-toggle=""tab"" class=""tabs"">Recurring Orders</a></li>'); });
</script>
```

API Plugin Development

Admin plugin's are not very useful without data and Znode API plugins are how plugin developers expose data. This section discusses the LogAPI which is an API plugin.

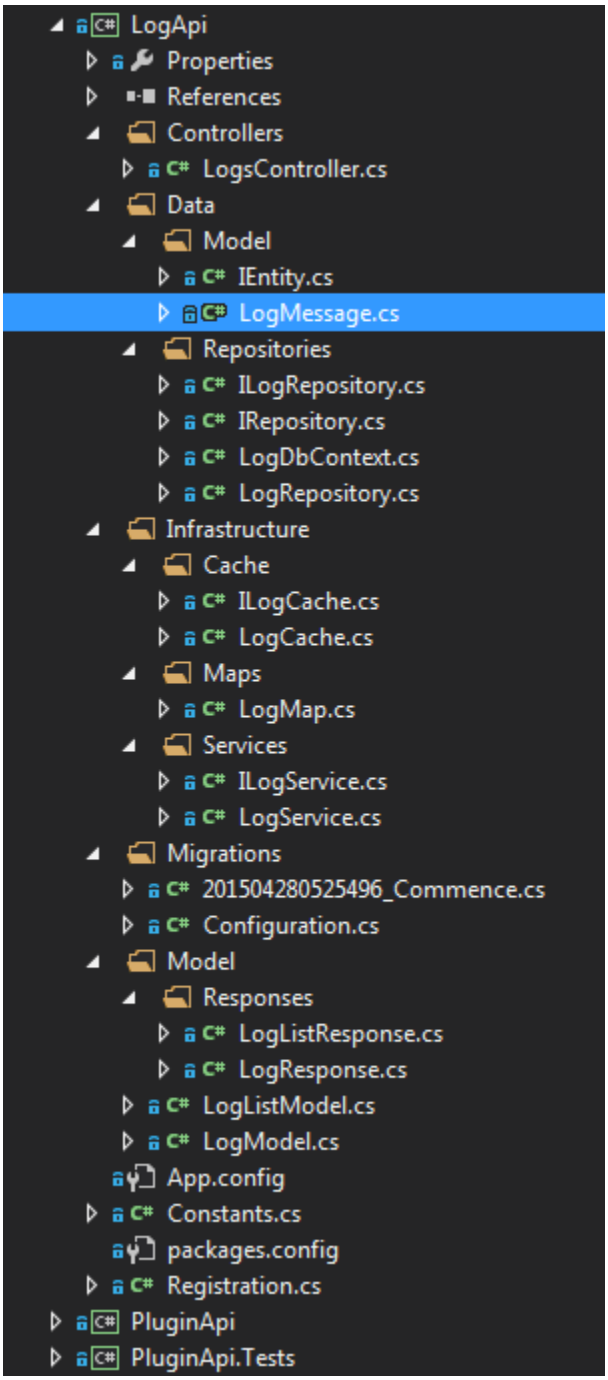
Project Settings

As with admin plugins, API plugins copy build output to their respective hosting process. Znode.Engine.API hosts API plugins and project build output settings should be set to copy build output into the plugins folder for Znode.Engine.API. LogAPI settings are shown below with the key elements highlighted. These setting should be set for All Configurations and Output Path for LogAPI is set to ..\..\Znode.Engine.API\Plugins\LogAPI\ where this path is a relative path from LogAPI to the host process Znode.Engine.API.



Project Structure

Project structure is a default ASP.NET MVC Class Library project, but the project follows conventions of ASP.NET WebAPI project.



References

Similar to admin plugin, API plugin references must match the location and version of the hosting project, Znode.Engine.API. Using NuGet to add references, can result in version conflicts between similarly named dlls.

Controllers

Controllers folders contains all API controllers for your API plugin. LogsController inherits from Znode.Framewor.API.Controllers.BaseController and this is a recommended practice to follow the API development patterns seen in Znode.Engine.API.

Data

Data folder contains Entity Framework Model, Repository, and DbContext. Znode API plugins are not constrained to using NetTiers.

Infrastructure

Infrastructure implements Cache, Services, and Mapping classes to be consistent with the Znode.Engine.API implementation.

Migrations

Migrations folder contains Entity Framework migrations to install, uninstall, and seed API plugin specific tables and database schema. Depending on your data access technology, API plugins implementations in this folder may differ.

Model

Model contains response contracts consisten with Znode.Engine.API patterns.

App.config

App.config is require when leveraging Entity Framework code first migrations. This is typically a development only artifact and is not required in the run-time environment. At run-time, API plugins will leverage database connection settings from Znode.Engine.API web.config file.

Registration.cs

This file implements Znode.API.Plugin.IAPIPlugin interface. This interface contains methods for installing and uninstalling your plugin. LogAPI shows an example of using Entity Framework migrations to install, seed, and uninstall database schema required by the LogAdmin API plugin.

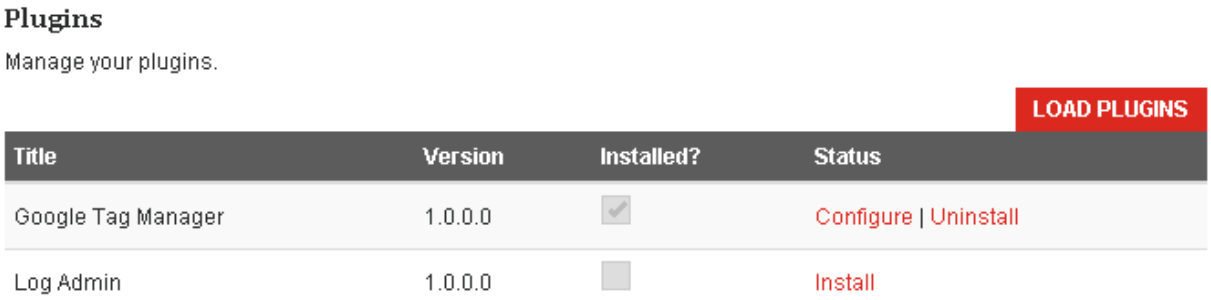
- IAPIPlugin.RegisterRoutes is where a plugin developer registers routes for an API plugin. LogAPI shows an example following Znode.Engine.API patterns.

Viewing API Plugin

To view results from LogAPI plugin, navigate to your website instance of Znode.Engine.API with the confgured route. In the instance of LogAdmin, an example url might be http://API.multifront.localhost.com/logs.

Load Plugins

Load plugin's is useful to load New Plugins on the server without stopping the sites.



Steps:

1. Paste the plugins dll's under the Plugins folder in Admin and Api
2. Hit the Load Plugins Button on Plugins in Admin.

Note:

For Load Plugins to work you have to give 'Network Sharing' and 'Everybody' permission to Global.asax file in Admin and Api. Otherwise you have to stop the site as well as the Application pool to load new plugins.