Shaper Quick Start

Quick Start guide for Shaper Middleware

- Introduction
  - What is Shaper
  - Credits
- Getting Started
  - Prerequisites
  - Development Environment

Introduction

What is Shaper

Shaper is a Middleware Framework built in C# and based on the multiplatform Microsoft ASP.NET Core.

Shaper responses to HTTP calls (REST, SOAP or anything you want) and executes one ore more procedures.

Shaper uses "application objects" to serve the requests:

- Tables
- Codeunits
- Pages
- Reports

Each of these objects "shape" the solution to obtain desired behaviour. Shaper has a local database to store and cache data; the database is arranged in "Tables". "Codeunits" are classes that cointains the business logic. "Pages" and "Reports" are the presentation layer.

Credits

Shaper by Brains is an idea of Simone Giordano (sg@simonegiordano.it)

License

Shaper is an Open Source project provided in dual licensing mode:

- Free for educational purposes and non commercial use
- Paid for commercial use (monthly support)

Contact us if you need a commercial license.

Getting Started

Prerequisites

Shaper is a multiplatform project.

For production or staging environment, you need only the latest "ASP.NET Core Runtime".

https://dotnet.microsoft.com/en-us/download/dotnet

For development, you need also the latest "Visual Studio".

https://visualstudio.microsoft.com/en-us/downloads/

If you want to use a local database, "Microsoft SQL Server" is recommended.

https://www.microsoft.com/en-us/sql-server/sql-server-downloads

Cloud or On Premise

Shaper is based on ASP.NET so you can host it where you want:

- On Premise with Windows and IIS
- On Premise with Linux and Nginx
- In a Docker environment
- On Microsoft Azure
- ...and so on!

Development Environment

Clone the "Shaper" main library from GitHub https://github.com/brayns-it/shaper

Create a new empty ASP.NET Core Project and simply call "InitializeShaper" and "MapShaperApi".

```
using Brayns.Shaper;

var builder = WebApplication.CreateBuilder(args);
builder.InitializeShaper();

var app = builder.Build();
app.MapShaperApi();

app.Run();
```

"MapShaperApi" will map two path in your web application:

- /api to serve REST request with GET, POST, PUT or DELETE method
- /rpc to serve special JSON request for the client

If you want to use also the web client (not only API) you have to clone "Shaper Web" library from GitHub https://github.com/brayns-it/shaper-web

Simply declare the web client and Web Sockets support:

```
app.MapShaperClient();
app.UseWebSockets();
```

"MapShaperClient" will catch all requests from "/client" base URI and serve the default index.html client page.

If you want to redirect also "/" path to default client page, add the following code:

```
app.MapShaperDefault();
```

Enable serving of static files and allows unknown MIME types (for example to enable Let's Encrypt HTTP validaton):

```
app.UseStaticFiles(new StaticFileOptions
{
    ServeUnknownFileTypes = true,
    DefaultContentType = "application/other"
});
```

To enable scheduled task:

```
app.UseShaperMonitor();
```

Mark the ASP.NET assembly as Shaper App container within the "AssemblyInfo" file (create it if doesn't exists):

```
[assembly: Brayns.Shaper.Classes.AppCollection]
```

Create the following directory structure:

- **code** (that contains specific project code)
- **var** (that contains configuration and logs)
- **var\resources** (that contains embedded resources)
- **wwwroot** (that contains HTTP served resources)

Complete "program.cs" example

```
using Brayns.Shaper;

var builder = WebApplication.CreateBuilder(args);
builder.InitializeShaper();

var app = builder.Build();
app.MapShaperApi();
app.MapShaperClient();
app.MapShaperDefault();
app.UseWebSockets();
app.UseStaticFiles(new StaticFileOptions
{
    ServeUnknownFileTypes = true,
    DefaultContentType = "application/other"
});
```

```
app.UseShaperMonitor();


app.Run();
```

Project Configuration

Project Configuration (csproj) must be adapted to:

- Embed PO files with translations
- Deploy **var\resources** folder

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  ...
  ...

  <ItemGroup>
    <None Remove="**/*.po" />
  </ItemGroup>

  <ItemGroup>
    <EmbeddedResource Include="**/*.po" />
  </ItemGroup>

  <ItemGroup>
    <None Update="var\resources\**">
      <CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
    </None>
  </ItemGroup>

  ...
  ...
</Project>
```

Publish Profile

Add the following lines to Publish Profile (pubxml) to preserve "var" directory:

```
<Project>
  ...
  ...

  <ItemGroup>
    <Content Update="var\resources" CopyToPublishDirectory="PreserveNewest" />
    <Content Update="var\**" CopyToPublishDirectory="Never" />
  </ItemGroup>

  ..
  ..
</Project>
```