StremioDotNet

StremioDotNet is an unofficial C# implementation of the Stremio Add-on SDK. It simplifies the creation and deployment of Stremio add-ons with an intuitive API and seamless integration with ASP.NET Core.

How to Download the StremioDotNet Package for .NET 6.0, 7.0, or 8.0

To download and install the `StremioDotNet` package for specific versions of .NET (6.0, 7.0, or 8.0), you can use either the NuGet command-line tool, Visual Studio, or the .NET CLI.

1. **Using the NuGet Command Line Tool**

Install the `StremioDotNet` Package

1. Open your terminal or command prompt.
2. To install the package, run the following command:

   ```
   nuget install StremioDotNet
   ```

   This will download the latest version of the package that is compatible with your environment.

Install a Specific Version for .NET

If you need to install a specific version of the `StremioDotNet` package for a particular version of .NET, you can use the `-Version` flag to specify the version of the package.

For example, to install version `1.0.0` of `StremioDotNet`, run:

```
nuget install StremioDotNet -Version 1.0.0
```

---

2. **Using the .NET CLI (dotnet)**

If you're working with .NET 6.0, 7.0, or 8.0, you can use the `dotnet add package` command to add `StremioDotNet` directly to your project.

Add the Package to Your Project

1. Open a terminal or command prompt.
2. Navigate to the directory of your .NET project.
3. Run the following command to add `StremioDotNet`:

```
dotnet add package StremioDotNet
```

Target a Specific Version of .NET

The `dotnet add package` command automatically resolves the correct version of the package for your project's target framework. However, if you want to ensure compatibility with a specific version of .NET (e.g., .NET 6.0, 7.0, or 8.0), you need to target that version in your project file.

You can check or set the target framework for your project in the `.csproj` file, like so:

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

  <PropertyGroup>
    <TargetFramework>net8.0</TargetFramework> <!-- or net7.0, net6.0 -->
  </PropertyGroup>

</Project>
```

Then, run:

```
dotnet restore
```

This will restore the dependencies for your project, including the `StremioDotNet` package, based on the target framework version.

---

3. **Using Visual Studio**

Add the `StremioDotNet` Package to Your Project

1. Open **Visual Studio**.
2. Right-click on your project in the **Solution Explorer** and select **Manage NuGet Packages**.
3. Go to the **Browse** tab and search for `StremioDotNet`.
4. Select the package and click **Install**.

Ensure your project is targeting `.NET 6.0`, `.NET 7.0`, or `.NET 8.0` in the **Target Framework** section under the **Properties** tab.

---

4. **Check Compatibility**

Make sure the version of `StremioDotNet` you're installing supports the version of .NET you're targeting. You can check the supported frameworks for the package on NuGet.org.

That's it! You've now installed the `StremioDotNet` package in your project, compatible with .NET 6.0, 7.0, or 8.0.

How to use

How to use StremioDotNet

Setup

This guide explains how to use the `StremioDotNet` package to build and configure Stremio addons.

Example Code

Below is an example of how to set up a `StremioDotNet` addon in an ASP.NET Core application:

```
using StremioDotNet.Builders;
using StremioDotNet.Extensions;
using StremioDotNet.Structs.Manifest;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddControllers();
builder.Services.AddStremio(); // Add Stremio

var app = builder.Build();

if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseRouting(); // Required, order matters!
app.UseHttpsRedirection();

AddonBuilder ConfigureAddon() => new AddonBuilder(
    "com.example",
    "1.0.0",
    "Example Addon",
```

```
    "Example Description",
    [AddonBuilder.Resources.Stream, AddonBuilder.Resources.Meta, AddonBuilder.Resources.Catalog],
    ["movie", "series"],
    ["tt", "ee"]
).SetCatalogs([
    new Catalog
    {
        Type = "movie",
        Id = "stremioDotNet",
        Name = "StremioDotNet Movies"
    }
]); // Configure Stremio

if (app.Environment.IsProduction())
{
    app.UseStremioAddon(() => ConfigureAddon().PublishToCentral("https://stremio.nathan.rip")); // Submit to
Stremio's repository in production
}
else
{
    app.UseStremioAddon(ConfigureAddon); // Local development
}

app.MapControllers();
app.Run();
```

Key Components

1. `AddStremio` Method

Definition:

```
public static void AddStremio(this IServiceCollection services)
```

- **Namespace**: `StremioDotNet.Extensions`
- **Purpose**: Registers the required services for a Stremio addon, including:
  - CORS configuration.

- An HTTP client.
- IMDb ID resolution middleware.

Usage:

```
builder.Services.AddStremio();
```

Parameters:

- `services`: The `IServiceCollection` to which the services are added.

---

2. `AddonBuilder` Class

Definition:

```
public class AddonBuilder
```

- **Namespace**: `StremioDotNet.Builders`

Purpose: The `AddonBuilder` class simplifies the creation and configuration of a Stremio Addon Manifest by providing:

- Methods to set behavior hints.
- Catalog definitions.
- Other necessary configurations.

Example:

```
AddonBuilder ConfigureAddon() => new AddonBuilder(
    "com.example",
    "1.0.0",
    "Example Addon",
    "Example Description",
    [AddonBuilder.Resources.Stream, AddonBuilder.Resources.Meta, AddonBuilder.Resources.Catalog],
    ["movie", "series"],
    ["tt", "ee"]
).SetCatalogs([
    new Catalog
    {
        Type = "movie",
        Id = "stremioDotNet",
```

```
        Name = "StremioDotNet Movies"
    }
]);
```

---

3. `UseStremioAddon` Method

Definition:

```
public static void UseStremioAddon(
    this IApplicationBuilder app,
    Func<AddonBuilder> createAddonBuilder)
```

- **Namespace**: `StremioDotNet.Extensions`
- **Purpose**: Configures the application to use a Stremio addon by:
  - Setting up middleware.
  - Mapping a GET endpoint (`/manifest.json`) to serve the addon manifest.

Usage:

```
app.UseStremioAddon(() => ConfigureAddon().PublishToCentral("https://stremio.nathan.rip"));
```

Parameters:

- `app`: The `IApplicationBuilder` instance used to configure the HTTP request pipeline.
- `createAddonBuilder`: A factory function that creates an `AddonBuilder` instance for manifest configuration.

Exceptions:

- `ArgumentNullException`: Thrown if `createAddonBuilder` is null.

Note: `PublishToCentral` makes a request to Stremio's repository of addons, allowing your addon to appear on the community addons page.

Summary

By using the `AddStremio`, `AddonBuilder`, and `UseStremioAddon` methods, you can quickly set up a fully functional Stremio addon within your ASP.NET Core application. For production environments, use `PublishToCentral` to submit your addon to Stremio's repository and make it visible in the community addons page.

StremioDotNet Documentation

This guide explains how to use the core features of the `StremioDotNet` library, including attributes, builders, predefined responses, and IMDb resolving middleware.

---

1. **Attributes**

Attributes in `StremioDotNet` simplify mapping methods to specific Stremio endpoints.

StreamHandler

Definition:

```
[StreamHandler(string type, bool resolveImdbId = false)]
```

Purpose:

Binds a method to handle stream requests in a Stremio service. It maps the method to an HTTP `GET` request with the URL pattern `stream/{type}/{id}.json`.

Parameters:

- **`type`** *(string, required)*: Specifies the type of stream (e.g., `movie`, `series`).
- **`resolveImdbId`** *(bool, optional, default: `false`)*: Indicates whether IMDb ID resolution is enabled.
- **`configType`** *(type, optional, default: `null`)*: The type of configuration to use for the route. If specified, the route will include a configuration parameter; otherwise, it will not.

Example Usage:

```
[StreamHandler("movie", true, typeof(ConfigExample))]
public IActionResult MovieHandler(string id, ConfigExample? config)
```

```
{
    var metadata = HttpContext.Items[id];
    if (metadata == null) return Streams();

    Console.WriteLine($"Access API Key: {config?.APIKey}");

    return Streams(new StreamBuilder()
        .SetName("Example Movie")
        .SetUrl("http://example.com/movie.mp4")
        .Build());
}

[Config]
public class ConfigExample
{
    [ConfigPropertyName("apikey")]
    public string? APIKey { get; set; }
}
```

CatalogHandler

Definition:

```
[CatalogHandler]
```

Purpose:

Maps a method to handle catalog requests. This attribute is bound to `GET catalog/{type}/{id}.json`.

Example Usage:

```
[CatalogHandler]
public IActionResult CatalogHandler(string type, string id)
{
    return Catalog(new MetaBuilder("example", type, "Catalog Example").Build());
}
```

MetaHandler

Definition:

```
[MetaHandler]
```

Purpose:

Maps a method to handle metadata requests. This attribute is bound to `GET meta/{type}/{id}.json`.

Example Usage:

```
[MetaHandler]
public IActionResult MetaHandler(string type, string id)
{
    return Meta(new MetaBuilder("example", type, "Meta Example").Build());
}
```

---

2. Builders

Builders in `StremioDotNet` are designed for constructing complex objects for Stremio responses.

StreamBuilder

Purpose:

Constructs a `Stream` object for `StreamHandler` responses.

Methods:

- `SetName(string name)`
- `SetUrl(string url)`
- `SetYtId(string ytId)`
- `SetDescription(string description)`
- `AddSubtitle(string id, string language, string url)`
- `SetInfoHash(string infoHash)`
- `SetFileIdx(int? fileIdx)`

- `SetBehaviorHints(bool? notWebReady, string bingeGroup, Dictionary<string, string> requestHeaders, Dictionary<string, string> responseHeaders)`
- `SetVideoSize(long? size)`
- `SetFilename(string filename)`

Example:

```
var streamBuilder = new StreamBuilder()
    .SetName("Example Stream")
    .SetUrl("http://example.com/stream.mp4")
    .SetBehaviorHints(notWebReady: false, bingeGroup: null, null, null)
    .Build();
```

MetaBuilder

Purpose:

Constructs a `Meta` object for `CatalogHandler` and `MetaHandler`.

Methods:

- `SetDescription(string description)`
- `SetGenres(IEnumerable<string> genres)`
- `SetPoster(string posterUrl)`
- `SetReleaseInfo(string releaseInfo)`
- `SetDirector(IEnumerable<string> directors)`
- `SetCast(IEnumerable<string> cast)`
- `SetTrailers(IEnumerable<Trailer> trailers)`
- `SetRuntime(string runtime)`
- `SetLanguage(string language)`
- `SetCountry(string country)`
- `SetAwards(string awards)`
- `SetWebsite(string website)`

Example:

```
var metaBuilder = new MetaBuilder("exampleId", "movie", "Example Movie")
    .SetDescription("A great movie.")
    .SetPoster("http://example.com/poster.png")
    .SetReleaseInfo("2024")
    .Build();
```

3. **Predefined Responses**

Streams

Returns a collection of streams for a content item.

```
return Streams(new StreamBuilder().SetName("Stream 1").SetUrl("http://example.com").Build());
```

Catalog

Returns a catalog of metadata items.

```
return Catalog(new MetaBuilder("catalogId", "movie", "Catalog Example").Build());
```

Meta

Returns metadata for a specific content item.

```
return Meta(new MetaBuilder("metaId", "movie", "Meta Example").Build());
```

---

4. **IMDb Resolving Middleware**

Automatically resolves IMDb IDs for requests.

Purpose:

Extracts IMDb IDs from request paths and resolves them into metadata, making it available in `HttpContext.Items`.

Setup:

```
builder.Services.AddStremio();
```

Example:

```
var metadata = HttpContext.Items["imdb_id"];
```

Summary

With `StremioDotNet`, creating Stremio addons is streamlined using:

- **Attributes** for mapping handlers.
- **Builders** for constructing responses.
- **Predefined responses** for common endpoints.
- **Middleware** for enriching requests with IMDb metadata.