

How to use

How to use StremioDotNet

- [Setup](#)
- [StremioDotNet Documentation](#)

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.