Step by Step Making Background Jobs Using Hangfire in net core

Overview

Hangfire is an open-source framework that An easy way to perform background processing in .NET and .NET Core applications. No Windows Service or separate process required.

Where is it used?

To background jobs with time for these scenarios:

- Send Email
- Send SMS
- maintaining DB
- batch import
- file(Video,Pic,..) processing
- Huge reports

Get latest source code here : https://github.com/abolfazlSadeqi/HangFireNetCoreTutorial

Steps

1. **Install NuGet Package(s) into the your Project:**

- Hangfire
- Hangfire.AspNetCore
- Hangfire.SqlServer

2.Create DB

```
CREATE DATABASE [HangFireNetCoreTutorial]
```

3. Add Configure the connection string into the appsettings.json file.

```
{
  "ConnectionStrings": {
    "HangfireConnection": "Password=123;Persist Security Info=True;User ID=sad;Initial
Catalog=HangFireNetCoreTutorial;Data Source=.;TrustServerCertificate=True;"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Warning",
      "Hangfire": "Information"
```

My Linkedin: https://www.linkedin.com/in/abolfazlsadeghi/ My Github: https://github.com/abolfazlSadeqi
My StackoverFlow: https://stackoverflow.com/users/10193401/abolfazl-sadeghi

```
    }
  }
}
```

4. **Add Configure things in Program.cs or Startup related to Hangfire, like SQL Server Database Connection and middleware**

```
builder.Services.AddHangfire(x => x.UseSqlServerStorage("<connection string>"));
builder.Services.AddHangfireServer();
```

5. Add Configure Hangfire dashboard in Program.cs or Startup

Monitoring UI allows you to see and control any aspect of background job processing, including statistics, exceptions and background job history.

```
    app.UseHangfireDashboard();

app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllerRoute(
             name: "default",
              pattern: "{controller=Reports}/{action=Index}/{id?}");

            // HangFire Dashboard endpoint
            endpoints.MapHangfireDashboard();
        });
```

Additional settings UseHangfireDashboard	
Read-only view	app.UseHangfireDashboard("/hangfire", new DashboardOptions { IsReadOnlyFunc = (DashboardContext context) => true });

Change URL Mapping	// Map the Dashboard to the root URL
	app.UseHangfireDashboard("");
	
	// Map to the `/jobs` URL
	app.UseHangfireDashboard("New Path");

Multiple Dashboards
 	
You can also map multiple dashboards that show information about different storages.
	You need to define a connection and add it to the dashboard
	a) define a connection
	var FistConnection = new SqlServerStorage("FistConnection");
	var SecendConnection = new SqlServerStorage("SecendConnection");
	
	b) add it to the dashboard
	app.UseHangfireDashboard("/hangfire1", new DashboardOptions(),FistConnection);
	app.UseHangfireDashboard("/hangfire2", new DashboardOptions(),SecendConnection);

My Linkedin: https://www.linkedin.com/in/abolfazlsadeghi/ My Github: https://github.com/abolfazlSadeqi
My StackoverFlow: https://stackoverflow.com/users/10193401/abolfazl-sadeghi

6) **HangfireBasicAuthMiddleware**

By default only local access is permitted to the Hangfire Dashboard. Dashboard authorization must be configured in order to allow remote access.
To make it secure by default, only local requests are allowed, however you can change this by passing your own implementations of the IDashboardAuthorizationFilter interface, whose Authorize method is used to allow or prohibit a request.

a)First,Add custom Attribute Authorization

```
public class HangFireNetCoreTutorialAuthorizationFilter : IDashboardAuthorizationFilter
{
    public bool Authorize(DashboardContext context)
    {
        var httpContext = context.GetHttpContext();

        // Allow all authenticated users to see the Dashboard (potentially dangerous).
       return httpContext.User.Identity?.IsAuthenticated ?? false;
    }
}
```

b)**Change UseHangfireDashboard(Add Attribute Authorization)**

```
app.UseHangfireDashboard("/hangfire", new DashboardOptions
{
   Authorization = new [] { new HangFireNetCoreTutorialAuthorizationFilter () }
});
```

7)Add IBackgroundJobClient or `IRecurringJobManager` **in Controller**

a)**For Fire-and-forget or Delayed Type or Continuations**

```
  private readonly IBackgroundJobClient _backgroundJobClient;
    public Fire_and_forgetController(IBackgroundJobClient backgroundJobClient)
    {
        _backgroundJobClient = backgroundJobClient;
    }
```

b)To Recurring Type

```
private readonly IRecurringJobManager _recurringJobManager;
    public RecurringController(IRecurringJobManager recurringJobManager)
    {
        _recurringJobManager = recurringJobManager;
    }
```

My Linkedin: https://www.linkedin.com/in/abolfazlsadeghi/ My Github: https://github.com/abolfazlSadeqi
My StackoverFlow: https://stackoverflow.com/users/10193401/abolfazl-sadeghi

Types of Jobs and Usage in Hangfire

- **Fire-and-forget :** These jobs are executed only once and almost immediately after they are fired.

Example(Send Email immediately)

```
_backgroundJobClient.Enqueue(() => Helper.SendMail(EmailType.Register));
```

- **Delayed :** Delayed jobs are executed only once too, but not immediately – only after the specified time interval.

Example(Send Email After 12 Hours)

```
_backgroundJobClient.Schedule(() => Helper.SendSMS(SMSType.Register), TimeSpan.FromHours(12));
```

- **Recurring :** Recurring jobs are fired many times on the specified CRON schedule.

Example(Backup Diff DB every Daily)

```
_recurringJobManager.AddOrUpdate("DiffBackupDB", () => Helper.DiffBackupDB(), Cron.Daily);
```

Type Cron: Minutely,Hourly,Daily,Weekly,Monthly,Yearly

- **Continuations :** Continuations are executed when parent job has finished.

Example(Send Email purchase After 45 Second after SendSMS purchase)

```
var jobId = BackgroundJob.Schedule(() => Helper.SendMail(EmailType.purchase), TimeSpan.FromSeconds(45));

BackgroundJob.ContinueJobWith(jobId, () => Helper.SendSMS(SMSType.purchase));
```

- **Batches :** Batch is a group of background jobs created atomically.
- **Batch Continuations :** Batch continuation is fired after all background jobs in a parent batch have finished.
- **Background Process :** Use them when you need to run background processes continuously throughout the lifetime of your application.

My Linkedin: https://www.linkedin.com/in/abolfazlsadeghi/ My Github: https://github.com/abolfazlSadeqi
My StackoverFlow: https://stackoverflow.com/users/10193401/abolfazl-sadeghi