Georgi Marokov

Posted on Nov 1, 2019 • Updated on <time datetime="2020-06-26T06:36:43Z" class="date">Jun 26, 2020</time> •
Originally published at worldwildweb.dev

Getting started with Hangfire on ASP.NET Core and PostgreSQL on Docker

#aspnet #hangfire #postgres #docker

Hangfire is an incredibly easy way to perform fire-and-forget, delayed and recurring jobs inside ASP.NET applications. No Windows Service or separate process required. Backed by persistent storage. Open and free for commercial use.

There are a number of use cases when you need to perform background processing in a web application:

- mass notifications/newsletter
- batch import from xml, csv, json
- creation of archives
- firing off web hooks
- deleting users
- building different graphs
- image/video processing
- purge temporary files
- recurring automated reports
- database maintenance

and counting..

We will get started by install and configure the database, then create new ASP.NET Core MVC project, after which we will get to Hangfire and run few background tasks with it.

Setup PostgreSQL database

There are more than one way to setup PostgreSQL database. I'm about to use Docker for the purpose, but you can install it directly from the Postgresql official webisite.

If you choose do download and install PostgreSQL, skip the following Docker commands.

Instead configure you db instance with the parameters from the Docker example.

Else we need Docker installed and running. Lets proceed with pulling the image for PostgreSQL. Open terminal and run:

```
$ docker pull postgresql
```

We have the image, let's create a container from it and provide username and password for the database:

```
$ docker run -d -p 5432:5432 -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=postgres
```

Create ASP.NET Core MVC project

So far we have the db up and running, continuing with the creation of the MVC project and configure it to use our database.

Create new folder and enter it:

```
$ mkdir aspnet-psql-hangfire && cd aspnet-psql-hangfire
```

When creating new project, you can go with whatever you want from the list of available dotnet project templates. I'll stick to mvc.

```
$ dotnet new mvc
```

Next install Nuget package for Entity Framework driver for PostgreSQL:

```
$ dotnet add package Npgsql.EntityFrameworkCore.PostgreSQL
```

Add empty dbcontext:

```
using Microsoft.EntityFrameworkCore;

namespace aspnet_psql_hangfire.Models
{
    public class DefaultDbContext : DbContext
    {
        public DefaultDbContext(DbContextOptions<DefaultDbContext> options)
            : base(options) { }
    }
}
```

Restore the packages by running:

```
$ dotnet restore
```

Edit `appsettings.json` and enter the connection string:

```
{
    "connectionStrings": {
        "defaultConnection":
            "Host=localhost;
            Port=5433;
            Username=postgres;
            Password=postgres;
            Database=aspnet-psql-hangfire-db"
    },
    "Logging": {
        "LogLevel": {
            "Default": "Warning"
        }
    },
    "AllowedHosts": "*"
}
```

The framework must know that we want to use PostgreSQL database so add the driver to your `Startup.cs` file within the ConfigureServices method:

```
services.AddEntityFrameworkNpgsql().AddDbContext<DefaultDbContext>(options => {
    options.UseNpgsql(Configuration.GetConnectionString("defaultConnection"));
});
```

We are ready for a initial migration:

```
$ dotnet ef migrations add InitContext && dotnet ef database update
```

Install Hangfire

Let's continue with final steps — install packages for Hangfire:

```
$ dotnet add package Hangfire.AspNetCore && dotnet add package Hangfire.Postgresql
```

Add the following using statement to the `Startup.cs`.

```
using Hangfire;
using Hangfire.PostgreSql;
```

Again in the ConfigureServices method in the `Startup.cs`, let Hangfire server to use our default connection string:

```
services.AddHangfire(x =>
    x.UsePostgreSqlStorage(Configuration.GetConnectionString("defaultConnection")));
```

Again in `Startup.cs`, but now in Configure method enter:

```
app.UseHangfireDashboard(); //Will be available under http://localhost:5000/hangfire"
app.UseHangfireServer();
```

Then restore again the packages by typing:

```
$ dotnet restore
```

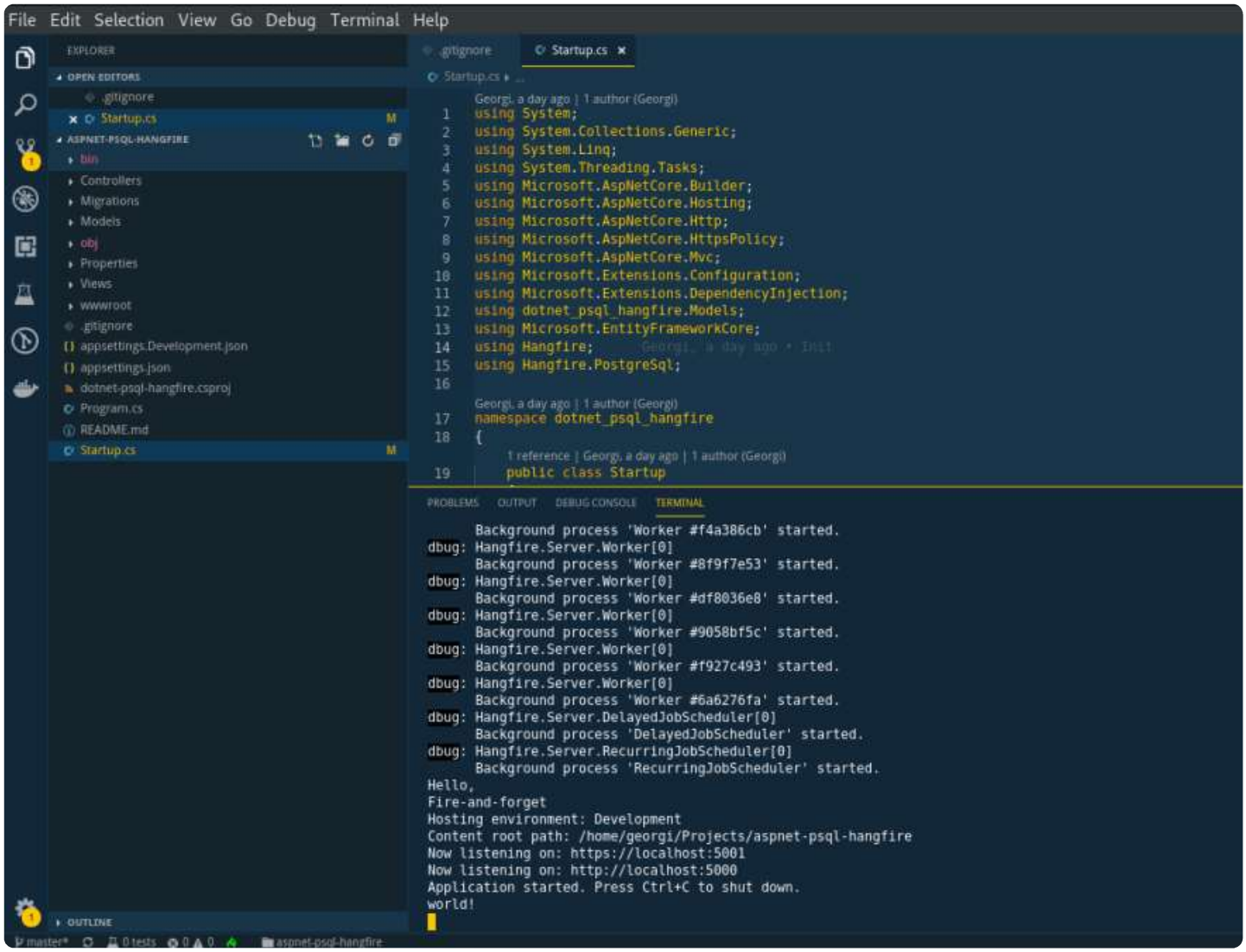
Create tasks

In the Configure method, below the `app.UseHangFireServer()` add the following tasks:

```
//Fire-and-Forget
BackgroundJob.Enqueue(() => Console.WriteLine("Fire-and-forget"));

//Delayed
BackgroundJob.Schedule(() => Console.WriteLine("Delayed"), TimeSpan.FromDays(1));

//Recurring
RecurringJob.AddOrUpdate(() => Console.WriteLine("Minutely Job"), Cron.Minutely);

//Continuation
var id = BackgroundJob.Enqueue(() => Console.WriteLine("Hello, "));
BackgroundJob.ContinueWith(id, () => Console.WriteLine("world!"));
```

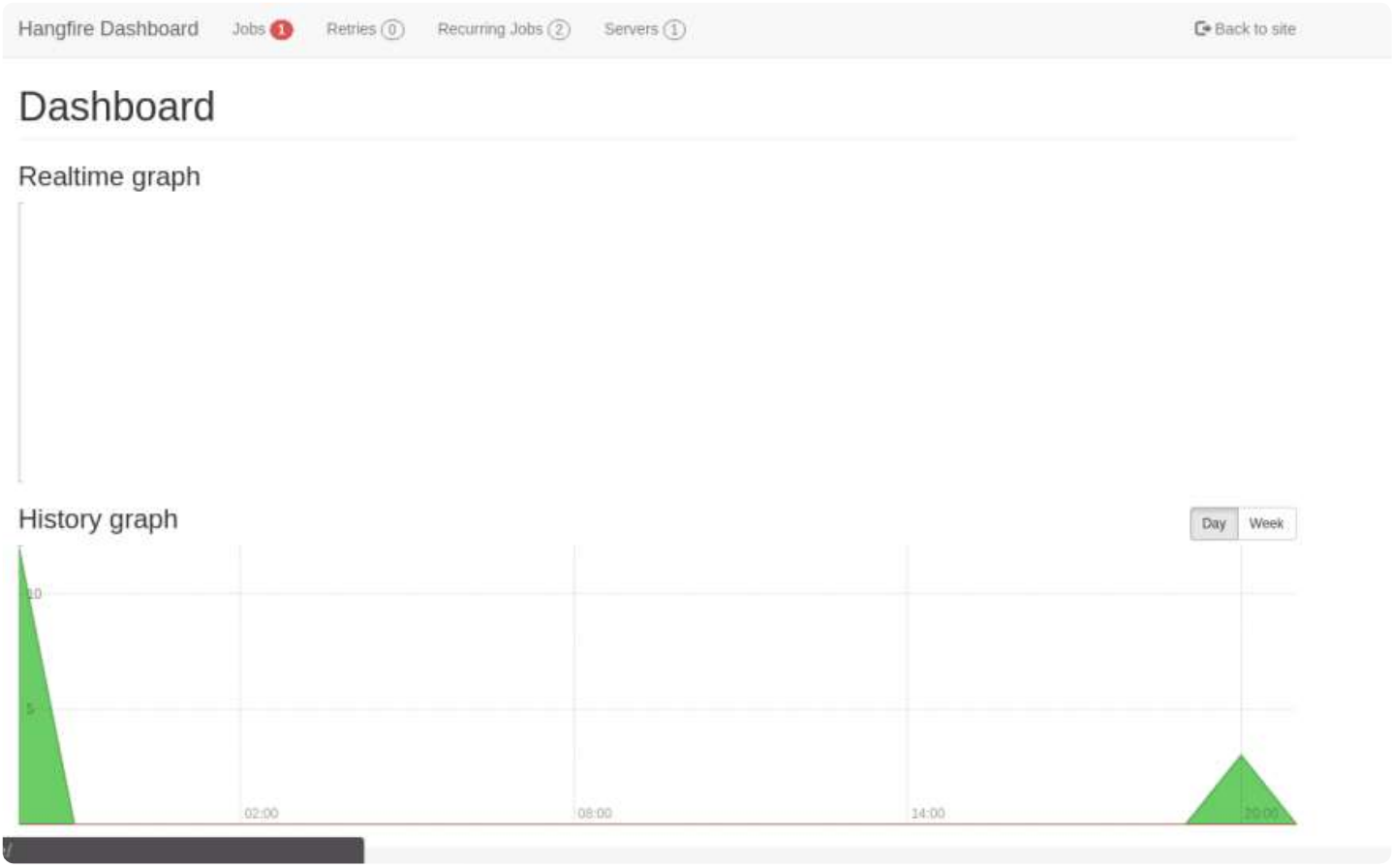
And finally run the app:

```
$ dotnet run
```



Observe the console.

Now go to the dashboard provided by Hangfire at http://localhost:5000/hangfire for more task info.



Summary

Keep in mind that the dashboard is only available for localhost connections. If you would like to use it in production, you have to apply authentication methods. There are plenty of tutorials describing how to do that.

Here is the repo from the project, I hope you liked it. Happy coding!

Discussion (2)

Tony Karalis • Jun 25 '20

Brilliant, thanks for the article.

Was there a reason for adding the NpgSql EF lib? I am guessing this is just for the sake of the example and not part of the hangifre config.

Also, small typo `app.UseHangFireServier() -> app.UseHangFireServer()`

Georgi Marokov • Jun 26 '20

Hey Tony! Thanks for the typo, it's fixed.
This example uses PostgreSQL as a data storage so using the driver lib is required. Also when configuring Hangfire you also need to specify the storage.
Of course that can be any other data storage you want :) Regards!

Georgi Marokov

FullStack developer, who is really into resolving complex issues, dive deep in to the problem and finding elegant solution. I spend my free time with family and friends, riding bikes and hiking peaks.

LOCATION
Sofia, Bulgaria

JOINED
Apr 29, 2019