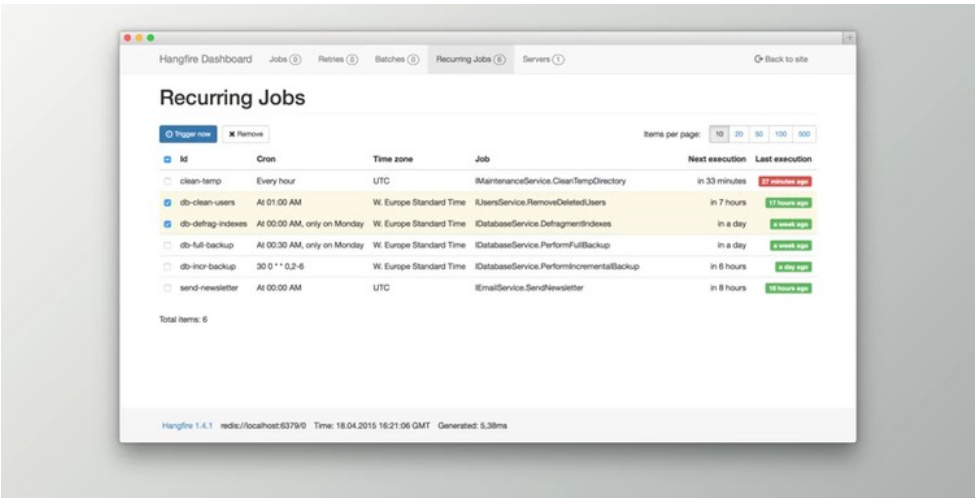
Hangfire Pro v3.0.5 and Hangfire Pro Redis v3.3.1 Retail

2026-07-06 20:14:59



Hangfire is an open-source platform that helps you to produce, process, and manage Your Own Desktop Tasks, i.e., Surgeries you do Not want to put in your request processing pipeline:

- Mass notifications/newsletter;
- batch import from XML, CSV, JSON;
- production of archives;
- shooting off-net hooks;
- deleting users;
- construction distinct charts;
- image/video processing;
- purge temporary documents;
- recurring automatic reports;
- database upkeep.

All Kinds of Background Jobs

Hangfire supports all sorts of background jobs -- short-running and long-running, CPU intensive and I/O intensive, 1 shot and continuing. You do not have to reinvent the wheel it is prepared to use.

Hangfire Guru is a group of extension packs that enhance performance and simplify the maintenance of desktop job processing in huge applications. Hangfire Pro packages are offered under paid subscriptions. After purchase, you get binaries, access to the personal NuGet feed and personal repository on GitHub.

Hangfire Pro and Hangfire Pro RedisGreat Features:

Instead of invoking a method synchronously, place it on a persistent queue, and the Hangfire worker thread will take it and perform within its own execution context:

This method creates a job in the storage and immediately returns control to the caller. Hangfire guarantees that the specified method will be called even after the abnormal termination of the host process.

Instead of invoking a method right now, you can postpone its execution for a specified time:

This call also saves a job, but instead of placing it in a queue, it adds the job to a persistent schedule. When the given time has elapsed, You will add the job to its queue. Meanwhile, you can restart your application – You will execute it anyway.

Hangfire uses persistent storage to store jobs, queues, and statistics and let them survive application restarts. The storage subsystem is abstracted enough to support both classic SQL Server and fast Redis.

- SQL Server provides simplified installation together with usual maintenance plans.
- Redis provides awesome speed, especially comparing to SQL Server, but requires additional knowledge.

If your method encounters a transient exception, don't worry – it will be retried automatically in a few seconds. If all retry attempts are exhausted, you can restart it manually from the integrated web interface.

You can also control the retry behavior with the AutomaticRetryAttribute class. Just apply it to your method to tell Hangfire the number of retry attempts:

Hangfire was made with the knowledge that the hosting environment can kill all the threads on each line. So, it does not remove the job until it is completed and contains different implicit retry logic to do the job when its processing was aborted.

所有软件资源 100%安全 已检测 无病毒 无篡改
去预览

标签

Other Components .Net

All the examples above use static method invocation, but instance methods are supported as well:

When a worker sees that the given method is an instance method, it will activate its class first. By default, the method is used, so only classes with default constructors are supported by default. But you can plug in your IoC container and pass the dependencies through the constructor.

When you marshal your method invocation into another execution context, you should preserve some environment settings. Some of them – `Thread.CurrentCulture` and `Thread.CurrentUICulture` are automatically captured for you.

It is done by the class that is applied to all of your methods by default.

Hangfire can tell your methods were aborted or canceled due to shutdown event, so you can stop them gracefully using job cancellation tokens that are similar to the regular `CancellationToken` class.

If you want to improve the testability of your job classes or don't want to use a huge amount of different factories, you should use instance methods instead of static ones. But you either need to pass the dependencies into these methods somehow, and the default job activator does not support parameter constructors.

Don't worry; you can use your favorite IoC container that will instantiate your classes. There are two packages, Hangfire.Ninject and Hangfire.Autofac for their respective containers. If you are using another container, please, write it yourself (based on the given packages) and contribute to the Hangfire project.

Hangfire uses the Common. Logging library to log all its events. It is a generic library, and you can plug it into your logging framework using adapters. Please, see the list of available adapters on NuGet Gallery.

You can run multiple Hangfire instances, either on the same or different machines. It uses distributed locking to prevent race conditions. Each Hangfire instance is redundant, and you can add or remove instances seamlessly (but control the queues they listen to).

Hangfire can process multiple queues. If you want to prioritize your jobs or split the processing across your servers (some processes the archive queue, others – the images queue, etc.), you can tell Hangfire about your decisions.

To place a job into a different queue, use the QueueAttribute class on your method:

To start to process multiple queues, you need to update your OWIN bootstrapper's configuration action:

The order is important; workers will fetch jobs from the critical queue first and then from the default queue.

Hangfire uses its own fixed worker thread pool to consume queued jobs. The default worker count is set to Environment.ProcessorCount * 5. This number is optimized both for CPU-intensive and I/O intensive tasks. If you experience excessive waits or context switches, you can configure the number of workers manually:

By default, the job processing is made within an ASP.NET application. But you can process jobs either in a console application, Windows Service, or anywhere else.

Hangfire is built to be as generic as possible. You can extend the following parts:

- storage implementation;
- states subsystem (including the creation of new states);
- job creation process;
- job performance process;
- state changing process;
- job activation process.

Some of core components are made as extensions: QueueAttribute, PreserveCultureAttribute, AutomaticRetryAttribute, SqlServerStorage, RedisStorage, NinjectJobActivator, AutofacJobActivator, ScheduledState.

This package includes the following items:

- Hangfire.Pro.2.3.1.nupkg
- Hangfire.Pro.2.3.1.with.Symbols.nupkg
- Hangfire.Pro.PerformanceCounters.2.3.1.nupkg
- Hangfire.Pro.PerformanceCounters.2.3.1.with.Symbols.nupkg
- Hangfire.Pro.Redis.2.8.16.nupkg
- Hangfire.Pro.Redis.2.8.16.with.Symbols.nupkg
- Hangfire.Pro.Redis.SEv2.2.8.16.nupkg
- Hangfire.Pro.Redis.SEv2.2.8.16.with.Symbols.nupkg
- Hangfire.Pro.Redis.StrongName.2.8.16.nupkg
- Hangfire.Pro.Redis.StrongName.2.8.16.with.Symbols.nupkg

版本列表

Hangfire Pro v2.2.1 and Hangfire Pro Redis v2.5.5 Retail

Hangfire Pro v2.2.4 and Hangfire Pro Redis v2.8.9 Retail

Hangfire Pro v2.3.0 and Hangfire Pro Redis v2.8.11 Retail

Hangfire Pro v2.3.1 and Hangfire Pro Redis v2.8.16 Retail

Hangfire Pro v2.3.2 and Hangfire Pro Redis v3.0.2 Retail

Hangfire Pro v3.0.5 and Hangfire Pro Redis v3.3.1 Retail

苏ICP备19032038号 Powered by getShopSys