layout: post title: Hangfire Tutorial (2) Job Types author: Andy Feng

Introduction

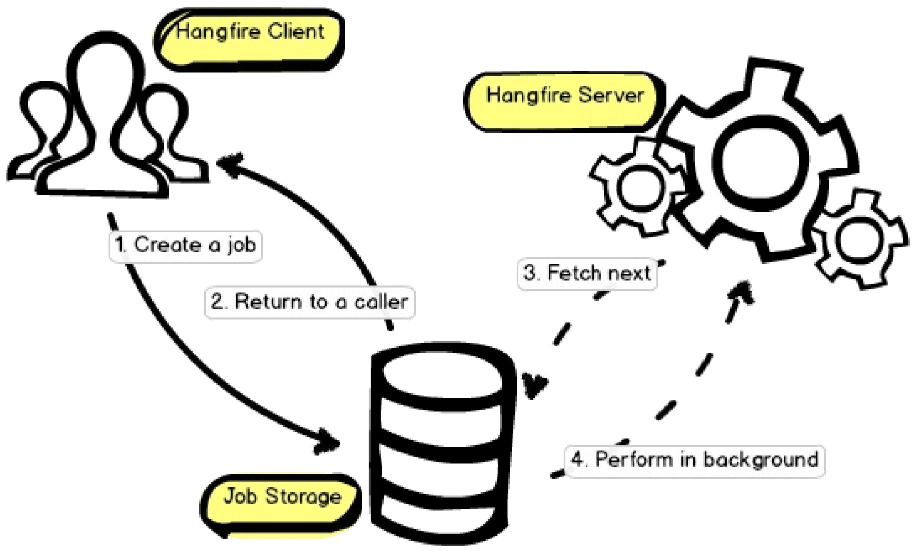
There are different types of background tasks. In previous section, we create a simple one-time-running task (namely fire-and-forget job). Here is a list of types of job.

Fire-and-forget job

Once you create a fire-and-forget job, it is saved to Hangfire queue. The queue is listened by a couple of dedicated workers that fetch a job and perform it. We simply call the Enqueue with the action we want to invoke on the server. i.e.

```
BackgroundJob.Enqueue(() => Console.WriteLine("I am a fire-and-forget job"));
```

Please note that the Enqueue method does not call the target method immediately. Here is the steps how it runs:



1. Caller enqueue a background job to Hangfire job storage
    - Caller create a fire-and-forget job via passing a lambda expression
    - Hangfire serialize a method information and all its arguments.
    - Hangfire create a new background job based on the serialized information.
    - Hangfire save background job to a persistent storage and enqueue it to queue.
    - Hangfire return to the caller
2. Hangfire execute jobs
    - Hangfire server picks up job and processes
    - Hangfire update job status

Delayed job

Delayed job is used in the senario when we want to delay the method invocation.. After the given delay, the job will be put to its queue and invoked as a regular fire-and-forget job. For instance, we can send an email to a user a couple days after they have signed up for a trial of your software.

```
BackgroundJob.Schedule(() => Console.WriteLine("I am a delayed job"),
TimeSpan.FromDays(1));
```

- Recurring job Recurring job is used to call a method on a recurrent basis (hourly, daily, etc) using the RecurringJob class. i.e.

```
RecurringJob.AddOrUpdate(() => Console.WriteLine("I am a daily recurring job"),
Cron.Daily);
```

Here, we can use CRON expressions to specify more complex scenarios.

Continuations job

Continuations job allow us to define complex workflows by chaining multiple background jobs together.

```
var id = BackgroundJob.Enqueue(() => Console.WriteLine("first job"));
BackgroundJob.ContinueWith(id, () => Console.WriteLine("second job"));
```

References

Workers patterns with hangfire