Top 20 Interview Questions and Answers on Background Jobs in .NET Core in 2025

Q1: What are background jobs in .NET Core?

Answer:
Background jobs are tasks that run asynchronously in the background, separate from the main request processing thread. They handle long-running, scheduled, or resource-intensive operations without blocking the user interface or request processing.
Example: Sending emails after user registration without making the user wait.

Q2: How do you implement background jobs in .NET Core?

Answer:
The primary built-in way is using IHostedService or inheriting from BackgroundService. You implement the ExecuteAsync method to define the background task.
Example:

```
public class MyBackgroundService : BackgroundService
{
    protected override async Task ExecuteAsync(CancellationToken stoppingToken)
    {
        while (!stoppingToken.IsCancellationRequested)
        {
            Console.WriteLine("Background job running...");
            await Task.Delay(1000, stoppingToken);
        }
    }
}
```

Q3: What is the difference between a hosted service and a regular service?

Answer:

- Hosted service (IHostedService) is designed to run background tasks with the application lifetime.
- Regular services are typically request-scoped or singleton services that serve API calls and are not designed for continuous background execution.

Q4: What are the different types of background jobs?

Ajay Saini

Answer:

- **Fire-and-forget:** Runs once immediately.
- **Delayed:** Runs once after a delay.
- **Recurring:** Runs repeatedly on a schedule.

Q5: What is Hangfire? How does it help with background jobs?

Answer:
Hangfire is a popular open-source framework that manages background jobs in .NET applications with persistent storage and a dashboard. It supports retries, scheduling, and recurring jobs.
Example:

```
BackgroundJob.Enqueue(() => Console.WriteLine("Hello from Hangfire!"));
```

Q6: How do you add Hangfire to a .NET Core project?

Answer:

1. *Install the NuGet package:* Install-Package Hangfire
2. *Configure in Program.cs:*

```
builder.Services.AddHangfire(config => config.UseSqlServerStorage("ConnectionString"));

builder.Services.AddHangfireServer();
```

3. *Use jobs:*

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

RecurringJob.AddOrUpdate("myjob", () => Console.WriteLine("Recurring job"), Cron.Daily);
```

Q7: How do you stop a background job gracefully?

Answer:
Use the CancellationToken provided in the ExecuteAsync method. The token is triggered when the app shuts down, allowing the job to complete or cancel gracefully.

Q8: What are the advantages of using Hangfire over IHostedService?

Answer:

- Job persistence with retry and failure tracking.
- Built-in dashboard to monitor jobs.
- Supports delayed and recurring jobs out of the box.
- Supports multiple storage backends.

Ajay Saini

Q9: What is Quartz.NET?

Answer:
Quartz.NET is an advanced open-source job scheduler for .NET, supporting complex scheduling like cron expressions, job chaining, and calendars.

Q10: When would you prefer Quartz.NET over Hangfire?

Answer:
When you need complex job scheduling with detailed timing (e.g., cron expressions, specific time zones), Quartz.NET is more flexible and powerful.

Q11: How can you schedule a recurring task with IHostedService?

Answer:
You can use a timer inside your hosted service to trigger code at intervals.
Example:

```
private Timer _timer;

protected override Task ExecuteAsync(CancellationToken stoppingToken)

{

    _timer = new Timer(DoWork, null, TimeSpan.Zero, TimeSpan.FromMinutes(5));

    return Task.CompletedTask;

}


private void DoWork(object state)

{

    Console.WriteLine("Running scheduled work");

}
```

Q12: Can background jobs in .NET Core survive application restarts?

Answer:

Ajay Saini

- IHostedService background jobs do not persist across restarts by default.
- Hangfire and Quartz.NET support persistence by saving jobs to storage, so they can resume after restart.

Q13: How do you monitor background jobs in Hangfire?

Answer:
Hangfire provides a web-based dashboard that shows job status, history, retries, and errors.

Q14: What are the challenges of background jobs?

Answer:

- Handling job failures and retries.
- Ensuring job persistence and idempotency.
- Managing concurrency and race conditions.
- Resource consumption and scaling.

Q15: How can you retry failed jobs in Hangfire?

Answer:
Hangfire automatically retries failed jobs. You can configure the retry count and delays.

Q16: Can you run multiple background jobs concurrently?

Answer:
Yes, hosted services run in their own threads/tasks and can run concurrently if designed properly. Hangfire and Quartz also support parallel execution.

Q17: What are some common use cases for background jobs?

Answer:

- Sending emails or notifications.
- Generating reports.
- Processing file uploads.
- Data cleanup tasks.
- Integration with external APIs.

Ajay Saini

Q18: What happens if a background job throws an exception?

Answer:

- In IHostedService, unhandled exceptions may stop the service or be logged.
- Hangfire retries jobs by default and logs failures.
- Quartz.NET can be configured for retries or failure handling.

Q19: How do you handle data access in background jobs?

Answer:
Inject your database services via Dependency Injection (DI) and ensure you handle context lifetimes properly, e.g., using scoped services inside background jobs.

Q20: Can you give an example of using BackgroundService with dependency injection?

Answer:

```
public class EmailBackgroundService : BackgroundService
{
    private readonly IEmailSender _emailSender;


    public EmailBackgroundService(IEmailSender emailSender)
    {
        _emailSender = emailSender;
    }


    protected override async Task ExecuteAsync(CancellationToken stoppingToken)
    {
        while (!stoppingToken.IsCancellationRequested)
        {
            await _emailSender.SendEmailsAsync();
            await Task.Delay(TimeSpan.FromMinutes(10), stoppingToken);
        }
    }
}
```

📢 **Feel free to repost, enhance, or adapt this content**

Ajay Saini