



Dot Net Full Stack Developer Interview Questions and Answers

Introduction

With our combined Dot and Full Stack course, students will get the opportunity to handle both front and back-end components of web applications using Microsoft's Dot Net Framework easily. This puts this skill in great demand currently. That is why we have curated these Dot Net Full Stack Interview Questions and Answers to give you a general knowledge of what sort of questions can be expected in Dot Net Full Stack Interview. So, go ahead and explore further to enhance your learning experience with our [Dot Net Full Stack Developer Course Syllabus](#).

List of Dot Net Full Stack Developer Interview Questions for Freshers

1. What is Dot Net Full Stack?

2. What is Entity Framework in Dot Net Full Stack?
3. What is dependency injection in Dot Net Full Stack?
4. What is ASP.NET Core?
5. What are the different types of Dependency Injection?
6. What are the main HTTP verbs used in ASP.NET Web API?
7. Explain the concept of Middleware in ASP Dot Net Core.
8. What are Razor Pages in ASP.NET Core?
9. Describe the role of NuGet in Dot Net development.
10. What are the different types of sessions in ASP.NET?

Reshape your career with our [dot net full stack developer course in Chennai](#).

Dot Net Full Stack Developer Interview Questions and Answers for Freshers

1. What is Dot Net Full Stack?

A Dot Full Stack Developer is skilled in both front-end (UI/UX design, using HTML, CSS, JavaScript, and frameworks like Angular or React) and back-end (server-side logic, database management with technologies such as C#, ASP.NET MVC/Core, and SQL databases). They deploy, maintain, and scale web applications using Microsoft's Dot Net framework, ensuring consistency, productivity, and scalability throughout the development lifecycle.

2. What is Entity Framework in Dot Net Full Stack?

Entity Framework (EF) is a Microsoft-provided ORM framework within the Dot Net ecosystem, streamlining the development of data-centric applications by enabling developers to interact with relational databases using object-oriented models. It supports multiple database providers, integrates with LINQ for efficient querying, and offers both Code-First and Database-First approaches for database schema management.

3. What is dependency injection in Dot Net Full Stack?

Dependency Injection (DI) in Dot Net Full Stack development is a pattern where dependencies of a component are provided externally, enhancing modularity and testability by decoupling components from their dependencies. It supports constructor, property, and method injection to manage dependencies, facilitated by

DI containers like Microsoft.Extensions.DependencyInjection in Dot Net Core/5/6, ensuring flexible and maintainable application architectures.

4. What is ASP.NET Core?

ASP.NET Core is a web framework that is open-source and tailored for creating modern, cloud-native, and internet-connected applications. It represents a revamped version of ASP.NET, incorporating architectural improvements aimed at boosting modularity, performance, and developer efficiency.

Take your Knowledge
Test Report

Check your Score



5. What are the different types of Dependency Injection?

Dependency Injection (DI) encompasses several forms:

- **Constructor Injection:** Dependencies are provided through a component's constructor.
- **Property Injection:** Dependencies are provided through public properties of the component.
- **Method Injection:** Dependencies are passed through method parameters.

6. What are the main HTTP verbs used in ASP.NET Web API?

The primary HTTP verbs employed in ASP.NET Web API are:

- **GET:** Retrieves data from the server.
- **POST:** Sends data to the server to create a new resource.
- **PUT:** Updates an existing resource on the server.
- **DELETE:** Removes a resource from the server.

7. Explain the concept of Middleware in ASP.NET Core.

Middleware refers to software components integrated into the ASP.NET Core pipeline. Each middleware component can handle incoming requests, pass them along to the next component, or intercept and respond to requests early in the pipeline flow.

8. What are Razor Pages in ASP.NET Core?

Razor Pages constitutes a page-focused development approach within ASP.NET Core for constructing web user interfaces. It merges the advantages of the MVC pattern with the simplicity of traditional web forms. Each Razor Page is composed of a .cshtml file containing embedded C# code to manage user requests.

9. Describe the role of NuGet in Dot Net development.

NuGet serves as a package manager designed for Dot Net that facilitates the discovery, installation, and management of libraries and tools. It simplifies the integration of third-party libraries into projects, handles dependency management effectively, and ensures packages are kept updated with minimal effort.

10. What are the different types of sessions in ASP.NET?

In ASP.NET, sessions can be managed through various methods:

- **In-Process Session:** Stores session data in the application process memory, offering fast access but limited to a single server instance.
- **State Server Session:** Utilizes the ASP.NET State Service (out-of-process) to store session data, allowing sharing across multiple servers.
- **SQL Server Session:** Stores session data in a SQL Server database, ensuring reliability and scalability at the expense of performance.

Kickstart your learning journey with our [dot net full stack developer tutorial for beginners](#).

List of Dot Net Full Stack Developer Interview Questions for Experienced

1. Differentiate Dot Net Core and Dot Net Framework.
2. What are the advantages of using Entity Framework?

3. How to ensure the security of web applications in Dot Net Full Stack?
4. How to optimize the performance of a Dot Net application?
5. How to manage state in Dot Net web application?
6. What are the ways to implement dependency injection in Dot Net Full Stack?
7. How does Dot Net support multi-threading and concurrency?

Explore our [Dot Net Full Stack Developer Salary in Chennai](#) for Freshers and Experienced Professionals.

Dot Net Full Stack Developer Technical Interview Questions and Answers for Experienced

1. Differentiate Dot Net Core and Dot Net Framework.

Aspect	Dot Net Framework	Dot Net Core
Design	Developed by Microsoft primarily for Windows	Open-source, modular framework for cross-platform use
Compatibility	Tightly integrated with Windows; supports C#, VB Dot Net, F#	Supports Windows, macOS, Linux; broader language support
Deployment	Requires a framework installed on the target machine	Can include runtime components for smaller deployments
Versioning	Long history (e.g., Dot Net Framework 4.x), backward compatible	Different versioning (e.g., Dot Net Core 1.x, 2.x, 3.x); unified with Dot Net 5+
Platform Support	Primarily for Windows	Cross-platform development
Deployment Efficiency	Larger deployment sizes	Smaller deployment sizes
Community and Evolution	Mature, closed-source; limited community involvement	Open-source encourages community contributions

Application Focus	Traditional Windows desktop and server applications	Modern cloud and containerized applications
--------------------------	---	---

2. What are the advantages of using Entity Framework?

Using Entity Framework (EF) in Dot Net Full Stack development provides several benefits:

- **Streamlined Data Access:** EF simplifies database interactions by using object-oriented models instead of direct SQL queries, enhancing development efficiency.
- **Enhanced Productivity:** EF automates CRUD operations, change tracking, and migrations, reducing the need for repetitive code and speeding up development.
- **Versatile Database Compatibility:** With its provider model, EF supports various databases like SQL Server, MySQL, and PostgreSQL, enabling easy transitions between platforms.
- **Integrated LINQ Queries:** EF integrates LINQ for type-safe queries in C# and VB.NET, improving code clarity and minimizing errors.
- **Flexible Development Approaches:** Supports both Code-First and Database-First approaches, accommodating different project needs and workflows.
- **Efficient Schema Management:** EF's migration feature allows incremental database updates, ensuring consistency across different application versions and environments.

3. How to ensure the security of web applications in Dot Net Full Stack?

Here are essential steps and practices to ensure the security of your Dot Net Full Stack web application:

Authentication and Authorization:

- Utilize powerful authentication mechanisms like Identity Framework or OAuth/OpenID Connect.

- Implement role-based or claims-based authorization to manage access rights effectively.

Secure Communication:

- Encrypt all client-server communications using HTTPS to safeguard sensitive data.

Input Validation and Sanitization:

- Validate and sanitize user inputs (e.g., forms, query parameters) to prevent XSS and SQL injection attacks.
- Use parameterized queries or stored procedures when interacting with databases to mitigate SQL injection risks.

Secure Authentication Tokens and Sessions:

- Configure cookies with HttpOnly and Secure flags to prevent client-side access.
- Implement secure token-based authentication (e.g. JWT tokens) with proper validation and storage practices.

4. How to optimize the performance of a Dot Net application?

The following are the ways to optimize the performance of a Dot Net application:

Use Performance Profiling Tools:

- Utilize tools like Visual Studio Profiler or JetBrains dotTrace to pinpoint bottlenecks in code, database queries, or external services.

Optimize Database Access:

- Improve query efficiency by minimizing round-trips, using indexes, and optimizing LINQ queries for Entity Framework or other ORMs.

Enhance Code Efficiency:

- Refactor to eliminate redundant code and leverage async/await for better responsiveness and resource management.

Implement Caching:

- Cache frequently accessed data using technologies like MemoryCache or Redis to reduce repetitive computations.

Optimize Network Communication:

- Reduce HTTP requests by combining resources and enabling compression (gzip, deflate) to minimize data transfer size.

5. How to manage state in Dot Net web application?

Managing state in a Dot Net web application involves handling and persisting data across user interactions and sessions. Common approaches include:

- ASP.NET Session State for server-side storage
- client-side options like Cookies, localStorage, and sessionStorage
- ASP Dot Net View State for page-level state retention
- ASP.NET Application State for application-wide data
- database solutions using SQL Server or NoSQL databases via Entity Framework
- Custom solutions, such as distributed caching or third-party libraries tailored to specific needs.

Take your Knowledge
Test Report

Check your Score



6. What are the ways to implement dependency injection in Dot Net Full Stack?

Steps to Implement Dependency Injection in Dot Net Full Stack:

- **Define Services and Dependencies:** Identify classes/interfaces representing services or dependencies that will be injected into other components, such as repositories or logging services.

- **Choose a DI Container:** Select a DI container framework supported by .NET, like Microsoft.Extensions.DependencyInjection, Autofac, Unity, Ninject, or Simple Injector, based on integration and support needs.
- **Configure DI Container:** Register services and their dependencies in the application's startup phase using methods like ConfigureServices in ASP Dot Net Core Startup class.
- **Inject Dependencies:** Use constructor, property, or method injection to inject services into components like controllers or services.
- **Automate Dependency Resolution:** Ensure the DI container automatically resolves dependencies during component instantiation, avoiding manual instantiation or service locators.
- **Test DI Configuration:** Validate DI setup with unit tests to confirm proper injection of dependencies, employing mocking frameworks for isolated testing.

7. How does Dot Net support multi-threading and concurrency?

Dot Net offers multiple mechanisms to support multi-threading and concurrency:

- **Threads:** Managed threads can be created using the System.Threading.Thread class.
- **Tasks:** The Task Parallel Library (TPL) simplifies parallelism, asynchronous programming, and task scheduling.
- **Async/Await:** This programming model enables non-blocking I/O operations, improving responsiveness.
- **Parallel LINQ (PLINQ):** Executes LINQ queries in parallel, leveraging multi-core processors for enhanced performance.

Conclusion

These **Dot Net Full Stack Interview Questions and Answers** are curated based on months of research and hard work from our team. These interview questions are some of the most frequently asked questions in Dot Net Full Stack interviews. By learning these interview questions, students will know what to expect from the interview, which eventually enhances their learning process. Kickstart or restart your career with our [software training institute in Chennai](#).