بالطبع، إليك إجابات الأسئلة:

1. ما الفرق بين .NET Framework و .NET Core؟

- **.NET Framework**:
  - مصمم لتطوير تطبيقات تعمل فقط على نظام التشغيل **ويندوز**.
  - غير مفتوح المصدر.
  - مناسب لتطبيقات سطح المكتب وتطبيقات الويب القديمة.
- **.NET Core**:
  - مفتوح المصدر ومتعدد المنصات (ويندوز، ماك، ولينكس).
  - يوفر أداءً أفضل ويدعم تطبيقات **Microservices** و **Cloud**.
  - يمكن تشغيله جنبًا إلى جنب مع .NET Framework على نفس الجهاز.

2. اشرح مفهوم CLR (Common Language Runtime) في .NET.

- **CLR** هو البيئة التنفيذية الأساسية في .NET.
  - يدير تنفيذ الكود المكتوب بـ C# أو VB.NET.
  - يقدم ميزات مثل:
    1. **Garbage Collection** لإدارة الذاكرة.
    2. **Exception Handling** لإدارة الأخطاء.
    3. **Type Safety** لضمان توافق أنواع البيانات أثناء التشغيل.

3. ما هو الفرق بين Value Type و Reference Type في C#؟

- **Value Type**:
  - تخزن القيمة مباشرة في الذاكرة المؤقتة (**Stack**).
  - يتم نسخ البيانات عند التمرير.
  - أمثلة: int, float, bool, struct.
- **Reference Type**:
  - تخزن عنوان الكائن في الذاكرة الديناميكية (**Heap**).
  - يتم تمرير المرجع بدلًا من نسخ البيانات.

- أمثلة: class, array, string.

4. ما هو مفهوم Garbage Collection وكيف يعمل في .NET؟

- **Garbage Collection (GC)** هو نظام إدارة تلقائية للذاكرة:
  - يقوم بتحرير الكائنات غير المستخدمة تلقائيًا.
  - يعمل على 3 أجيال (**Generations**):
    1. **Gen 0**: الكائنات الجديدة.
    2. **Gen 1**: الكائنات التي نجت من Gen 0.
    3. **Gen 2**: الكائنات طويلة الأمد.

5. ما هو الـ NuGet؟ وكيف يُستخدم في مشاريع .NET؟

- **NuGet** هو مدير حزم في .NET:
  - يسمح بإضافة مكتبات خارجية إلى المشروع بسهولة.
  - يتم تثبيت الحزم عن طريق **Visual Studio** أو **CLI** باستخدام الأمر:

```
Install-Package <PackageName>
```

6. اشرح مفهوم Assemblies و Namespaces.

- **Assemblies**:
  - ملفات تنفيذية أو مكتبات (.exe أو .dll).
  - تحتوي على الكود المترجم وعناصر المشروع.
- **Namespaces**:
  - تُستخدم لتنظيم الكود ومنع تضارب الأسماء بين الكلاسات.
  - مثال:

```
using System;
namespace MyApp {
    class Program {
        static void Main() {
            Console.WriteLine("Hello, World!");
        }
    }
}
```

7. ما هو الفرق بين Managed Code و Unmanaged Code؟

- **Managed Code**:
  - يتم تشغيله بواسطة **CLR**.
  - يوفر ميزات مثل إدارة الذاكرة وتأمين الكود.
- **Unmanaged Code**:
  - يتم تشغيله مباشرة بواسطة النظام.
  - يتطلب إدارة يدوية للذاكرة.
  - أمثلة: الكود المكتوب بلغة **C++**.

8. اذكر أنواع الـ Access Modifiers في C# مع شرح لكل نوع.

1. **Public**:
   - متاح لجميع الكلاسات والمشاريع.
2. **Private**:
   - متاح فقط داخل نفس الكلاس.
3. **Protected**:
   - متاح داخل نفس الكلاس والكلاسات المشتقة.
4. **Internal**:
   - متاح داخل نفس المشروع فقط.
5. **Protected Internal**:
   - متاح داخل نفس المشروع أو الكلاسات المشتقة.

9. ما الفرق بين IEnumerable و IQueryable؟

- **IEnumerable**:
  - يتم تحميل البيانات كلها في الذاكرة.
  - مناسب للاستعلامات داخل الذاكرة (**In-Memory**).
- **IQueryable**:
  - يتم تنفيذ الاستعلام على قاعدة البيانات مباشرة.
  - يُستخدم في استعلامات كبيرة الحجم لتحسين الأداء.

10. اشرح مفهوم Delegates و Events مع مثال عملي.

- **Delegates**:
  - مؤشر يشير إلى دالة.
  - يمكن استدعاء الدالة من خلال الـ Delegate.

مثال:

```
public delegate void PrintMessage(string message);
class Program {
    static void ShowMessage(string msg) {
        Console.WriteLine(msg);
    }
    static void Main() {
        PrintMessage pm = ShowMessage;
        pm("Hello, Delegates!");
    }
}
```

- **Events**:
  - تُستخدم لإرسال إشارات عند حدوث شيء معين.
  - تعتمد على الـ Delegates.

مثال:

```
public delegate void Notify();
public class Process {
    public event Notify OnComplete;
    public void StartProcess() {
        Console.WriteLine("Process Started");
        OnComplete?.Invoke();
    }
}
class Program {
    static void Main() {
        Process process = new Process();
        process.OnComplete += () => Console.WriteLine("Process
Completed");
        process.StartProcess();
    }
}
```

إجابات أسئلة ASP.NET Core (Intermediate):

1. ما هي Middlewares في ASP.NET Core؟ وكيف تعمل؟

- **Middlewares** هي مكونات تُستخدم لمعالجة الطلبات والاستجابات في تطبيق ASP.NET Core.
- قبل الوصول إلى Middlewares حيث يمر كل طلب من خلال سلسلة من الـ "Pipeline" تعمل كسلسلة أو المعالج النهائي.

مثال:

```
public void Configure(IApplicationBuilder app) {
    app.Use(async (context, next) => {
                                                    // قبل معالجة الطلب
        await next.Invoke(); // استدعاء المكون التالي
                                                    // بعد معالجة الطلب
    });
    app.UseStaticFiles(); // Middleware للملفات الثابتة
    app.UseRouting(); // Middleware لعملية التوجيه
}
```

2. اشرح مفهوم Dependency Injection وكيف يتم تطبيقه في ASP.NET Core.

- **Dependency Injection** هو نمط تصميم يساعد على تقليل الاعتمادية بين الكلاسات.
- يتم تطبيقه باستخدام **Service Container** لإدارة الكائنات، في ASP.NET Core

مثال:

```
                                                    // تسجيل الخدمة
services.AddTransient<IMyService, MyService>();

                                                    // استخدام الخدمة
public class HomeController {
    private readonly IMyService _service;
    public HomeController(IMyService service) {
        _service = service;
    }
}
```

3. ما هو الفرق بين MVC و Razor Pages؟

- **MVC** (Model-View-Controller: يعتمد على ثلاث طبقات ويفصل منطق العمل عن العرض.
- **Razor Pages**: يعمل كصفحة واحدة تحتوي على الكود والواجهة معًا، مما يجعلها أبسط للتطبيقات الصغيرة.

4. ما هو Entity Framework Core؟ وكيف يُستخدم في إدارة قواعد البيانات؟

- **Entity Framework Core** هو إطار عمل ORM يُستخدم للتعامل مع قواعد البيانات باستخدام الكود بدلًا من الاستعلامات اليدوية.

مثال:

```
public class ApplicationDbContext : DbContext {
    public DbSet<Product> Products { get; set; }
}
```

5. اشرح مفهوم Routing في ASP.NET Core.

- **Routing** يحدد كيفية تعيين طلبات URL) إلى المعالجات (Handlers مثل Controllers أو Actions أو Razor Pages.
- يتم تعريفه باستخدام MapControllerRoute أو Endpoints.

مثال:

```
app.UseRouting();
app.UseEndpoints(endpoints => {
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");
});
```

6. ما هي Filters في MVC؟ وما أنواعها؟

- **Filters** هي مكونات تُستخدم لتحديد منطق يُنفذ قبل أو بعد تنفيذ Action.
- الأنواع:
  1. **Authorization Filters**: للتحقق من الصلاحيات.
  2. **Resource Filters**: تنفيذ منطق قبل تشغيل الـ Action وبعده.
  3. **Action Filters**: تشغيل منطق قبل وبعد تنفيذ الـ Action.
  4. **Exception Filters**: للتعامل مع الأخطاء.
  5. **Result Filters**: تنفيذ منطق قبل وبعد تنفيذ النتيجة.

7. كيف تُنفّذ Authentication و Authorization في ASP.NET Core؟

- **Authentication**: التحقق من هوية المستخدم باستخدام الـ Middleware.
- **Authorization**: تحديد الصلاحيات للمستخدمين المعتمدين.

مثال بسيط:

```
services.AddAuthentication("CookieAuth").AddCookie("CookieAuth", options
=> {
    options.LoginPath = "/Account/Login";
});

services.AddAuthorization(options => {
    options.AddPolicy("AdminOnly", policy => policy.RequireRole("Admin"));
});
```

8. ما الفرق بين AddSingleton و AddScoped و AddTransient في الـ Dependency Injection؟

- **AddSingleton**: يتم إنشاء كائن واحد يُستخدم طوال عمر التطبيق.

- **AddScoped**: يتم إنشاء كائن جديد لكل طلب HTTP.
- **AddTransient**: يتم إنشاء كائن جديد لكل استدعاء.

9. ما هو الفرق بين REST API و SOAP؟

- **REST API**:
  - يعتمد على HTTP ويستخدم JSON أو XML.
  - خفيف وسهل الاستخدام.
- **SOAP**:
  - بروتوكول يعتمد على XML.
  - يدعم ميزات مثل **Security** و **Transactions**.

10. كيف تقوم برفع مشروع ASP.NET Core إلى IIS أو Azure؟

- **IIS**:
  1. قم بنشر المشروع باستخدام **Publish** في Visual Studio.
  2. انسخ الملفات إلى مجلد في IIS وأضف موقعًا جديدًا.
- **Azure**:
  1. استخدم خيار "Publish to Azure" من Visual Studio.
  2. اختر الاشتراك وقم بإنشاء App Service جديد.

إجابات أسئلة متقدمة (Advanced):

1. اشرح مفهوم Microservices وكيف يتم تطبيقه باستخدام .NET.

- **Microservices** هي بنية تصميم تُقسم التطبيق إلى خدمات صغيرة مستقلة يمكن نشرها وإدارتها بشكل منفصل.
- تُطبق باستخدام ASP.NET Core APIs وDocker لتشغيل الخدمات في حاويات.

2. ما هو SignalR؟ وكيف يُستخدم لتطبيقات الوقت الفعلي؟

- **SignalR** إطار عمل لتطبيقات الوقت الفعلي مثل الدردشة.
- يُوفر اتصالاً مستمرًا بين العميل والخادم باستخدام WebSockets أو تقنيات أخرى.

مثال:

```
services.AddSignalR();
app.UseEndpoints(endpoints => {
    endpoints.MapHub<ChatHub>("/chatHub");
});
```

3. كيف تُحسن أداء تطبيقات .NET Core؟

- استخدم **Caching** لتحسين الأداء.
- نفّذ الكود بشكل غير متزامن باستخدام async/await.
- استخدم Load Balancing لتوزيع الحمل.

4. ما الفرق بين Synchronous و Asynchronous Programming؟

- **Synchronous**: يتم تنفيذ المهام واحدة تلو الأخرى.
- **Asynchronous**: يتم تنفيذ المهام بالتوازي دون الانتظار.

مثال:

```
public async Task<string> GetDataAsync() {
    var data = await httpClient.GetStringAsync("https://example.com");
    return data;
}
```

5. كيف تعمل Unit Testing في .NET باستخدام xUnit أو MSTest؟

- **xUnit**: إطار عمل لاختبار الوحدات.

مثال:

```
[Fact]
public void Test_Addition() {
    Assert.Equal(4, 2 + 2);
}
```

6. ما هو مفهوم CQRS؟

- **CQRS (Command Query Responsibility Segregation)**: هو نمط يفصل بين منطق القراءة والكتابة لتوفير أداء أفضل.

7. كيف تتعامل مع الأخطاء باستخدام Global Exception Handling؟

- أضف Middleware مخصص:

```
app.UseExceptionHandler(errorApp => {
    errorApp.Run(async context => {
        var error = context.Features.Get<IExceptionHandlerFeature>();
        await context.Response.WriteAsync("Error occurred");
    });
});
```

8. ما هو Docker وكيف يمكن استخدامه مع .NET Core؟

- **Docker**: يُستخدم لحزم وتشغيل التطبيقات في حاويات.

- لإنشاء صورة:

```
FROM mcr.microsoft.com/dotnet/aspnet:6.0
COPY . /app
WORKDIR /app
ENTRYPOINT ["dotnet", "App.dll"]
```

9. اشرح مفهوم Polymorphism في C# مع أمثلة.

- **Polymorphism** يعني وجود أكثر من شكل للوظيفة.
- يمكن تطبيقه باستخدام **Overloading** أو **Overriding**.

مثال:

```
public class Animal {
    public virtual void Speak() => Console.WriteLine("Animal speaks");
}
public class Dog : Animal {
    public override void Speak() => Console.WriteLine("Dog barks");
}
```

10. كيف يتم استخدام Identity Server لتطبيقات الـ OAuth/OpenID Connect؟

- يُستخدم Identity Server لإدارة تسجيل الدخول والصلاحيات باستخدام OAuth 2.0 و OpenID Connect.
- يتم دمجه مع ASP.NET Core لتقديم خدمات Authentication.