ASP.NET Core - True Ultimate Guide

Section 23: SOLID Principles - Notes

SOLID Principles

SOLID is an acronym representing five key principles of object-oriented design:

1. **Single Responsibility Principle (SRP)**
    - A class should have only one reason to change.
    - Promotes focused and maintainable classes.
2. **Open/Closed Principle (OCP)**
    - Software entities should be open for extension but closed for modification.
    - Encourage adding new features without changing existing code.
3. **Liskov Substitution Principle (LSP)**
    - Objects of a derived class should be substitutable for objects of the base class without affecting the correctness of the program.
    - Ensures that inheritance relationships are used appropriately.
4. **Interface Segregation Principle (ISP)**
    - Clients should not be forced to depend on interfaces they do not use.
    - Promotes smaller, more focused interfaces.
5. **Dependency Inversion Principle (DIP)**
    - High-level modules should not depend on low-level modules. Both should depend on abstractions.
    - Abstractions should not depend on details. Details should depend on abstractions.
    - Encourages loose coupling and flexibility.

Benefits of SOLID Principles

- **Maintainability:** Makes your code easier to understand, modify, and extend.
- **Testability:** Promotes writing unit tests by encouraging loose coupling and dependency injection.
- **Flexibility:** Makes your code adaptable to changes in requirements.
- **Reusability:** Encourages the creation of reusable components.

Interview Tips

- **Understanding:** Be able to explain each principle clearly and concisely.
- **Examples:** Provide real-world or code examples that demonstrate how to apply each principle.
- **Benefits:** Articulate the advantages of adhering to SOLID principles.
- **Trade-offs:** Acknowledge that there might be trade-offs and complexities in applying these principles in certain situations.
- **Practical Application:** Discuss how you have used or would use SOLID principles in your own projects.

Example Code (Conceptual)

```
// SRP (Single Responsibility Principle)
public class ProductService
{
   // Handles product-related logic, like adding or retrieving products.
}

public class OrderService
{
   // Handles order-related logic, like creating or processing orders.
}



// OCP (Open/Closed Principle)
public interface IPaymentProcessor
{
   void ProcessPayment(PaymentDetails details);
}



public class CreditCardPaymentProcessor : IPaymentProcessor { /* ... */ }
public class PayPalPaymentProcessor : IPaymentProcessor { /* ... */ }
```

```
// LSP (Liskov Substitution Principle)
public class Rectangle
{
    public virtual int Width { get; set; }
    public virtual int Height { get; set; }
    // ...
}


public class Square : Rectangle
// Violates LSP
{
    public override int Width
    {
        get => base.Width;
        set
        {
            base.Width = value;
            base.Height = value; // Setting width also sets height
        }
    }

    public override int Height
    {
        get => base.Height;
        set
        {
            base.Height = value;
            base.Width = value; // Setting height also sets width
        }
```

```
    }
}


// ISP (Interface Segregation Principle)
public interface IPrinter
{
    void Print();
}

public interface IScanner
{
    void Scan();
}

public class PrintScanMachine : IPrinter, IScanner { /* ... */ }

// DIP (Dependency Inversion Principle)
public class OrderProcessor
{
    private readonly IPaymentProcessor _paymentProcessor;

    public OrderProcessor(IPaymentProcessor paymentProcessor)
    {
        _paymentProcessor = paymentProcessor;

    }

    // ...
}
```