



---

### ## 3. Strategy

\*\*  
\*\*

```csharp

```
public interface IPaymentStrategy { void Pay(); }
public class AlipayStrategy : IPaymentStrategy { public void Pay() { /*...*/ } }
public class WeChatPayStrategy : IPaymentStrategy { public void Pay() { /*...*/ } }
```

```
public class PaymentContext
{
    private readonly IPaymentStrategy _strategy;
    public PaymentContext(IPaymentStrategy strategy) { _strategy = strategy; }
    public void ExecutePay() => _strategy.Pay();
}
```

```
//
services.AddTransient<IPaymentStrategy, AlipayStrategy>();
services.AddTransient<PaymentContext>();
```
```

---

### ## 4. Decorator

\*\*  
\*\* AOP

```csharp

```
public class LoggingRepository<T> : IRepository<T>
{
    private readonly IRepository<T> _inner;
    public LoggingRepository(IRepository<T> inner) { _inner = inner; }
    public void Add(T entity) { Console.WriteLine(""); _inner.Add(entity); }
}
```

```
//
services.Decorate(typeof(IRepository<>), typeof(LoggingRepository<>)); // Scrutor
```
```

---

### ## 5. Observer

\*\*  
\*\*

```csharp

```
public interface IEventHandler<T> { void Handle(T evt); }
```



---

## 8. Proxy

\*\*`RepositoryProxy` implements `IRepository`

```
```csharp
public class RepositoryProxy<T> : IRepository<T>
{
    private readonly IRepository<T> _inner;
    public RepositoryProxy(IRepository<T> inner) { _inner = inner; }
    public void Add(T entity) { /* ... */ _inner.Add(entity); }
}
// DI
services.Decorate(typeof(IRepository<>), typeof(RepositoryProxy<>));
```
```

---

## 9. Command

\*\*`CommandInvoker` implements `ICommandInvoker`

```
```csharp
public interface ICommand { void Execute(); }
public class CreateOrderCommand : ICommand { /* ... */ }
public class CommandInvoker
{
    private readonly IEnumerable<ICommand> _commands;
    public CommandInvoker(IEnumerable<ICommand> commands) { _commands = commands; }
    public void ExecuteAll() { foreach (var cmd in _commands) cmd.Execute(); }
}
// DI
services.AddTransient<ICommand, CreateOrderCommand>();
```
```

---

## 10. Template Method

\*\*`BaseModule` implements `IModule`

```
```csharp
public abstract class BaseModule
{
    public void RegisterModule()
    {
```

```

    PreRegister();
    DoRegister();
    PostRegister();
}
protected virtual void PreRegister() { }
protected abstract void DoRegister();
protected virtual void PostRegister() { }
}
...

```

---

## 11. Builder

\*\*[Builder](#)    \*\*[Builder](#)    SQL    Http

```

```csharp
public class DbContextBuilder
{
    private string _connStr;
    public DbContextBuilder UseSqlServer(string connStr) { _connStr = connStr; return this; }
    public MyDbContext Build() => new MyDbContext(_connStr);
}
...

```

---

## 12. Flyweight

\*\*[Flyweight](#)    \*\*[Flyweight](#)

```

```csharp
public class DbContextPool
{
    private readonly Dictionary<string, MyDbContext> _pool = new();
    public MyDbContext Get(string connStr)
    {
        if (!_pool.ContainsKey(connStr))
            _pool[connStr] = new MyDbContext(connStr);
        return _pool[connStr];
    }
}
...

```

---

## 13. Composite





## ## 20. 工厂方法 Prototype

\*\*工厂方法\*\*      \*\*工厂方法

```
```csharp
public interface IPrototype<T> { T Clone(); }
public class ConfigPrototype : IPrototype<ConfigPrototype>
{
    public string Value;
    public ConfigPrototype Clone() => (ConfigPrototype)this.MemberwiseClone();
}
```
```

---

## ## 21. 链式调用 Pipeline/Filter

\*\*链式调用\*\*      \*\*链式调用

```
```csharp
public interface IPipelineStep { void Process(RequestContext ctx); }
public class AuthStep : IPipelineStep { public void Process(RequestContext ctx) { /*...*/ } }
```
```

---

## ## 22. 门面 Facade

\*\*门面\*\*      \*\*门面

```
```csharp
public class PaymentFacade
{
    private readonly IPaymentStrategy _alipay;
    private readonly IPaymentStrategy _wechat;
    public PaymentFacade(IPaymentStrategy alipay, IPaymentStrategy wechat)
    {
        _alipay = alipay; _wechat = wechat;
    }
    public void PayByAlipay() => _alipay.Pay();
    public void PayByWeChat() => _wechat.Pay();
}
```
```

---

## ## 23. 数据库访问 DAO





