Fluent API

Dot NET COP

2024/11/21

Agenda

1. Overview of Fluent API
2. Pros and cons of using Fluent APIs
3. Use cases of Fluent APIs
4. Overview of Builder Pattern
5. Code Demo

What is Fluent API?

Wikipedia

“relies extensively on **method chaining**”

“increase **code legibility** by creating a domain-specific language”

Fowler’s Fluent Interface article

“primarily designed to be **readable** and to **flow**”

“**chaining** is a common technique to use with fluent interfaces, but true fluency is much more than that.”

“The more the use of the API has that **language like flow**, the more fluent it is.”

What is Fluent API?

A Fluent API is set of interfaces or methods in which the methods can be **chained** one after another **returning a valid object** and can be read in **language like flow** making it more **readable** and **discoverable** for the users.

Examples

```
public class Product
{
    [Key]
    public int ProductId { get; set; }

    [Required]
    [MaxLength(100)]
    public string Name { get; set; }

    [Column(TypeName = "decimal(18,2)")]
    public decimal Price { get; set; }
}
```

Defining database constraints in model class using Data Annotations for EF Core

Examples

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Product>(entity =>
    {
        entity.HasKey(p => p.ProductId);
        entity.Property(p => p.Name)
              .IsRequired()
              .HasMaxLength(100);
        entity.Property(p => p.Price)
              .HasColumnType("decimal(18,2)");
    });
}
```

Defining database constraints using Fluent API for EF Core

Examples

```
builder.Services.AddAuthentication(options => {
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
})
    .AddJwtBearer(options =>
    {
        options.RequireHttpsMetadata = false;
        options.TokenValidationParameters = new TokenValidationParameters
        {
            ValidateAudience = false,
            ValidateIssuer = false,
            ValidateIssuerSigningKey = true,
            IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("<JWT Secret>")),
            ClockSkew = TimeSpan.Zero,
        };
    });
```

Adding JWT Authentication using Fluent API

Benefits of Using Fluent API

1. Encapsulates domain-specific operations
2. Provides better code completion and IntelliSense
3. Self documenting code
4. More concise and readable code
5. Enhances developer experience

Cons of Using Fluent API

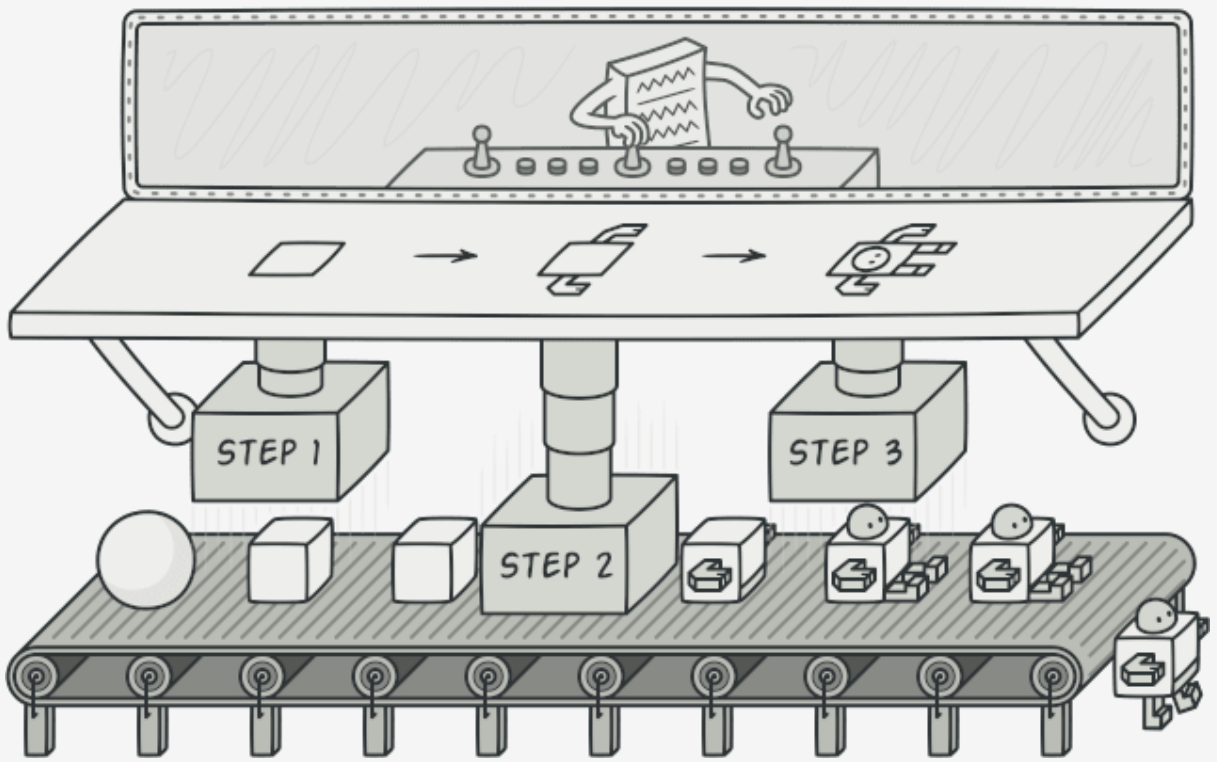
1. Requires more thoughtful decisions
2. Violates the principle of Command Query Separation
3. Methods do not make sense on their own

Use cases

1. Domain-specific languages
2. Configurable objects
3. Defining long lived utilities

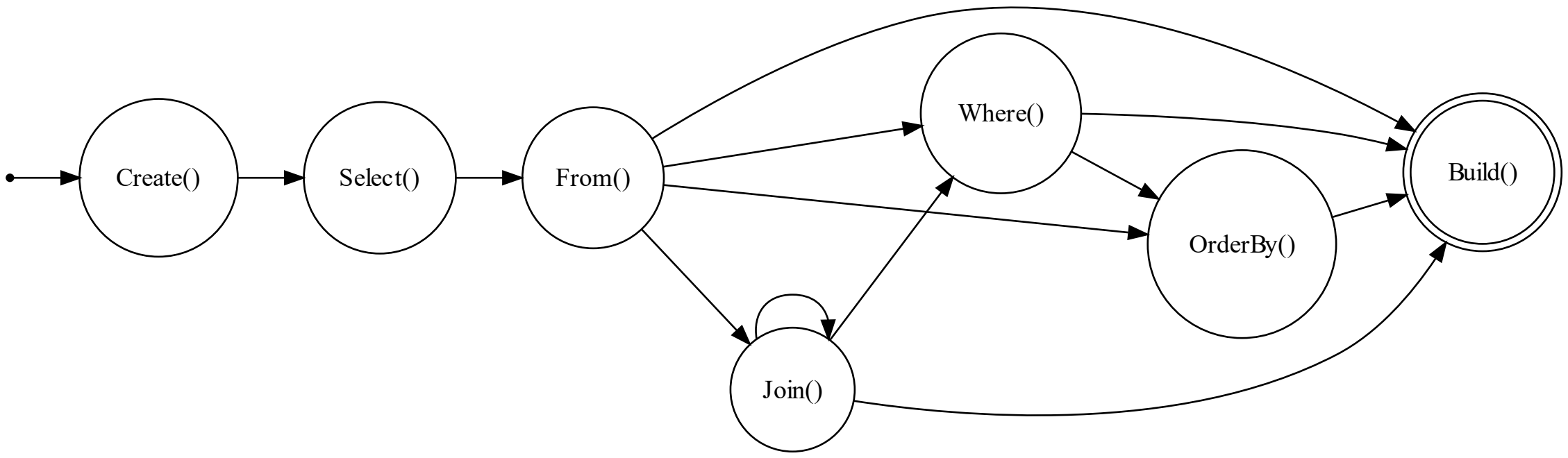
Builder Pattern

- Creational design pattern
- Allows to create complex objects step by step
- Allows to produce different types and representations of an object using the same construction code

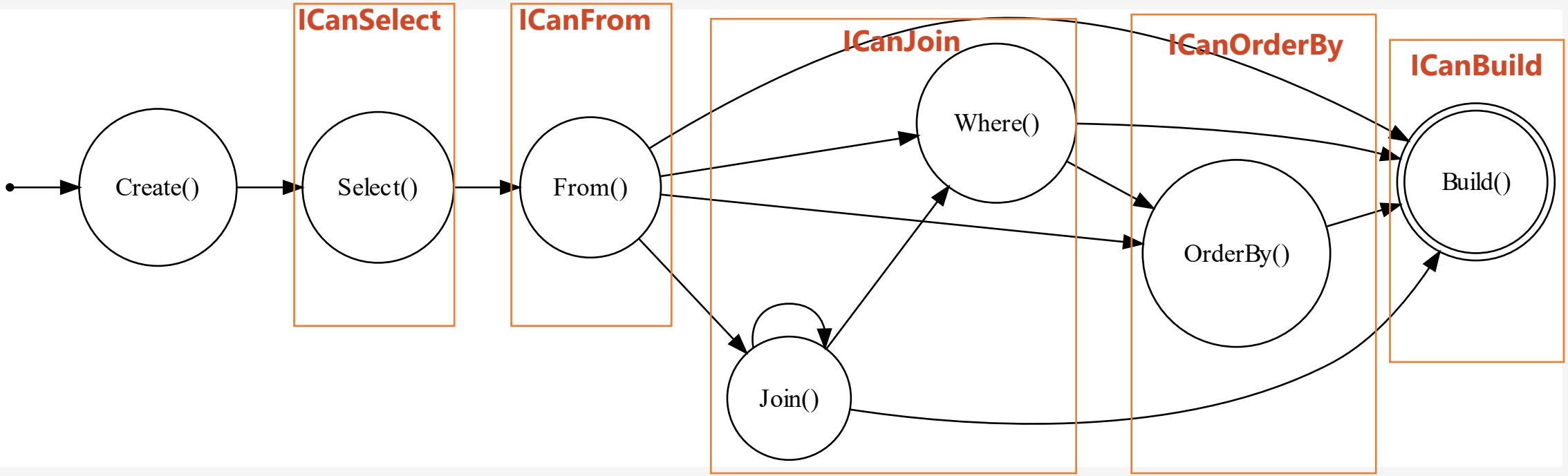


Coding Demo

Demo: QueryBuilder



Demo: QueryBuilder



Any Questions?

Thank You