

Introducing an Anemic Domain Model

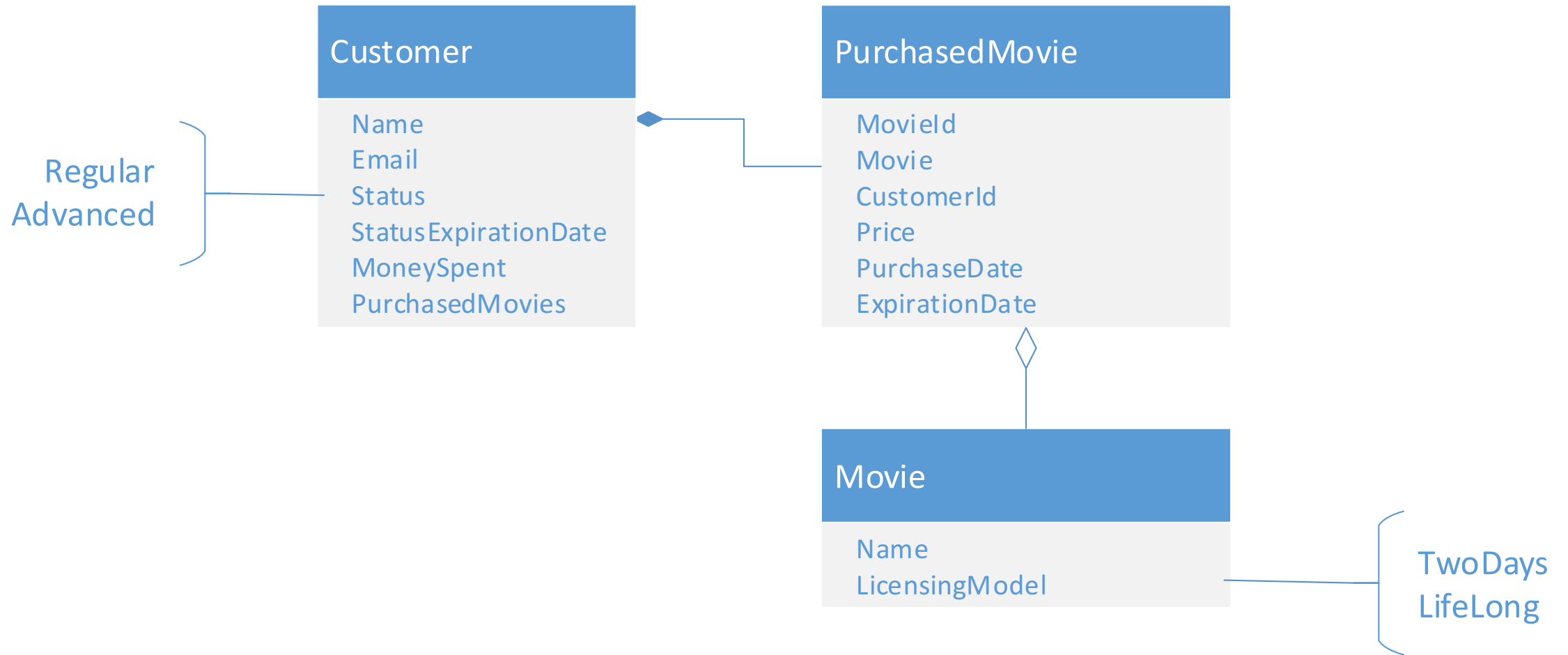


Vladimir Khorikov

@vkhorikov www.enterprisecraftsmanship.com



Domain Model Introduction



Functionality



- Purchasing a movie



- Create a customer
- Update a customer



- Provide discounts



- Retrieve a list of all customers
- Retrieve a detailed information about a customer




Application Code Introduction

<http://bit.ly/anemic-code>



Application Code Introduction

 vkhorikov / **AnemicDomainModel**

Unwatch 1

Star 0

Fork 0

<> Code

Issues 0

Pull requests 0

Projects 0

Wiki

Settings

Insights

Source code for the Anemic Domain Model Pluralsight course

Edit

[Add topics](#)

3 commits

1 branch

0 releases

1 contributor

MIT

Branch: master


New pull request

Create new file

Upload files

Find file

Clone or download

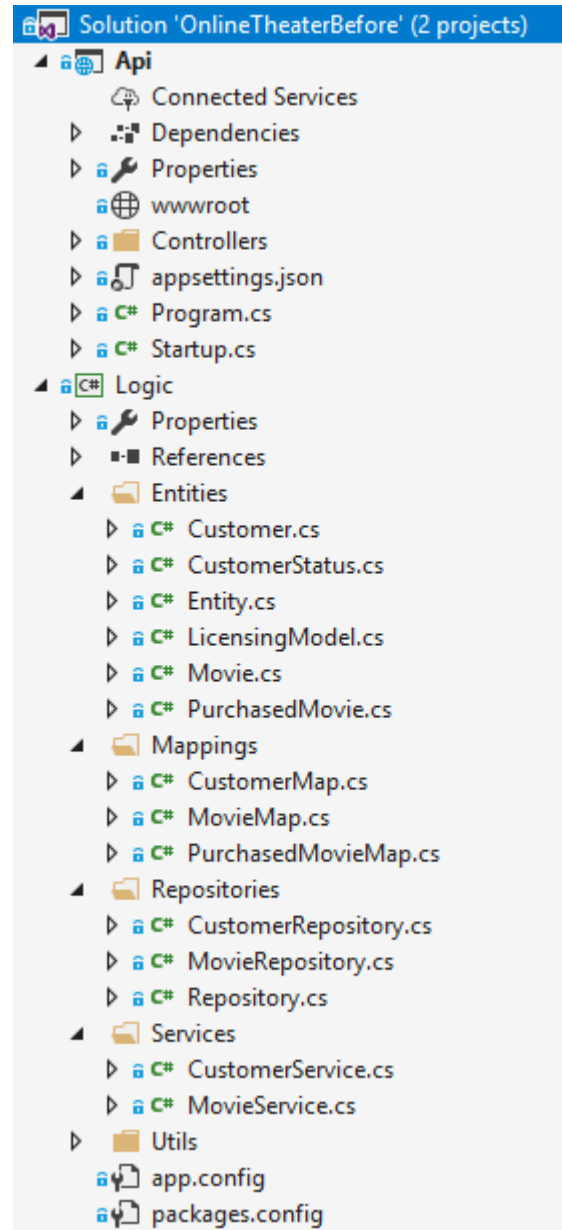
 vkhorikov **Readme**

Latest commit 27b26e3 6 days ago

After	initial comit	6 days ago
Before	initial comit	6 days ago
.gitignore	initial comit	6 days ago
LICENSE	Create LICENSE	6 days ago
README.md	Readme	6 days ago



Application Code Drawbacks



Application Code Drawbacks

```
public class Customer : Entity {  
    [Required, MaxLength(100, ErrorMessage = "Name is too long")]  
    public virtual string Name { get; set; }  
  
    [Required, RegularExpression(@"^(.+)?@(.+)$", ErrorMessage = "Email is invalid")]  
    public virtual string Email { get; set; }  
  
    [JsonConverter(typeof(StringEnumConverter))]  
    public virtual CustomerStatus Status { get; set; }  
    public virtual DateTime? StatusExpirationDate { get; set; }  
    public virtual decimal MoneySpent { get; set; }  
    public virtual IList<PurchasedMovie> PurchasedMovies { get; set; }  
}
```



Poor discoverability



Lack of encapsulation



Application Code Drawbacks

```
customer.Status = CustomerStatus.Advanced;  
customer.StatusExpirationDate = DateTime.UtcNow.AddYears(1);
```

```
customer.Status = CustomerStatus.Advanced;  
customer.StatusExpirationDate = null;
```

```
customer.Status = CustomerStatus.Advanced;  
customer.StatusExpirationDate = DateTime.UtcNow.AddYears(-1);
```



Application Code Drawbacks

```
public class Customer : Entity {  
    [Required, MaxLength(100, ErrorMessage = "Name is too long")]  
    public virtual string Name { get; set; }  
  
    [Required, RegularExpression(@"^(.+)(.+)@", ErrorMessage = "Email is invalid")]  
    public virtual string Email { get; set; }  
  
    [JsonConverter(typeof(StringEnumConverter))]  
    public virtual CustomerStatus Status { get; set; }  
    public virtual DateTime? StatusExpirationDate { get; set; }  
    public virtual decimal MoneySpent { get; set; }  
    public virtual IList<PurchasedMovie> PurchasedMovies { get; set; }  
}
```



Nothing prevents you from
violating the invariants

Make sure to expose the
absolute minimum of
operations that you can execute
upon your domain model.



Application Code Drawbacks

```
public class Customer : Entity {  
    [Required, MaxLength(100, ErrorMessage = "Name is too long")]  
    public virtual string Name { get; set; }  
  
    [Required, RegularExpression(@"^(.+)?@(.+)$", ErrorMessage = "Email is invalid")]  
    public virtual string Email { get; set; }  
  
    [JsonConverter(typeof(StringEnumConverter))]  
    public virtual CustomerStatus Status { get; set; }  
    public virtual DateTime? StatusExpirationDate { get; set; }  
    public virtual decimal MoneySpent { get; set; }  
    public virtual IList<PurchasedMovie> PurchasedMovies { get; set; }  
}
```



Summary



Application code introduction

Anemic domain model

Lack of encapsulation



In the Next Module

Decoupling the Domain Model from Data Contracts

