HYF .NET Masterclass 2022

Week 4

Agenda for today

- Recap on Week #3 and homework
- A quick look at Model-View-Controller in a real life web site
- Interfaces
- Service classes and Dependency Injection
- Mealsharing

Homework recap

- Enums – what are they and how do they work?
- Returning list of objects – or IEnumerables
- Discussion – what would be the best approach for a converter?

Real-life MVC CMS example

- Models are content types
- Editorial interface
- Example is Optimizely (Episerver) CMS – but Umbraco would be very similar

Interfaces

Describes certain characteristics (Signatures) on a class.

If a class implements that interface, it must implement everything described in the interface

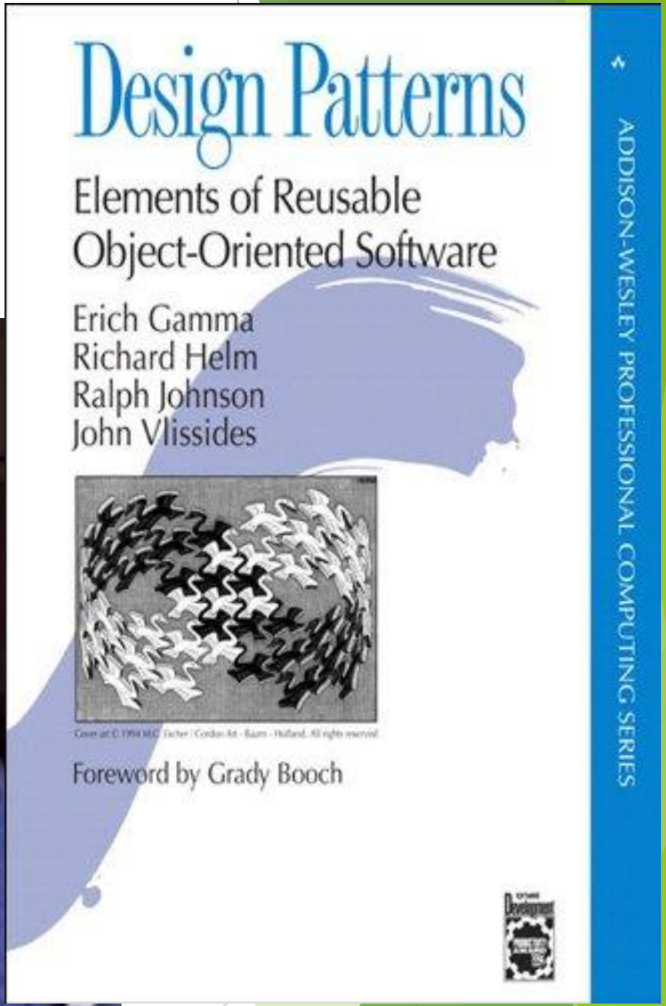
Any class can implement any number of interfaces

Often used as a substitute for multiple inheritance

Naming standard: Start with “I”

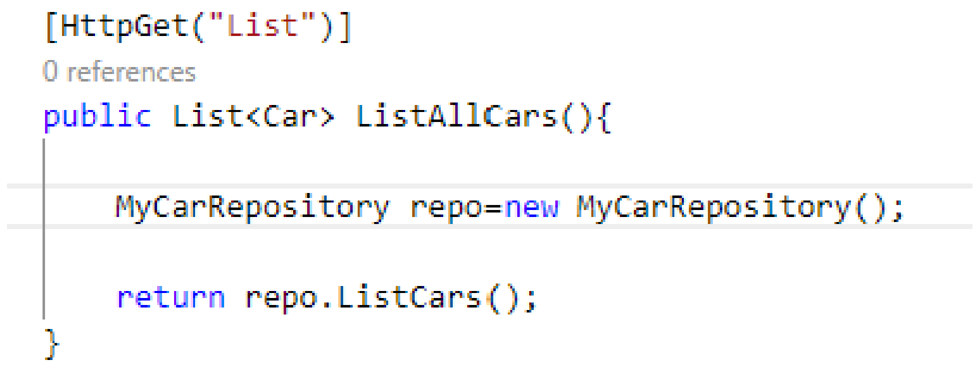
Demo

Design Pattern: Inversion Of Control


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

IoC: What is the problem

- Business logic should be separated out into re-usable classes



```
[HttpGet("List")]
0 references
public List<Car> ListAllCars(){

    MyCarRepository repo=new MyCarRepository();

    return repo.ListCars();
}
```

- But the above will make a ‘hard’ dependency on that specific business logic class, MyCarRepository.
- If there in the future is another MyCarRepository that should be used, you will also have to change it here.

IoC problem cont’d

Problem if your code is being used as a library – nothing can easily be replaced in it without full source code.

Problem for Unit testing if you want to use Mock versions of business logic

Solution: Inversion of Control

- Have ‘something’ provide the correct implementation for any requested type/interface dynamically.
- For example using “Service Locator” pattern
- Or use “Dependency Injection” to have it injected in the classes that need it.
- Standard with Dependency Injection in ASP.NET Core.

Register *services* in Startup or Program

Builder.Services.AddTransient
Transient services are created when needed and disposed instantly

Builder.Services.AddScoped
Scoped services are created when needed, but kept throughout the current request

Builder.Services.AddSingleton
Singletons are kept as long as the web application is running. *Singleton* is also a 'Pattern' for a class of which there is only one implementation which is never disposed.

Use *services* in controller

- Constructor injection
- Service Locator
- Injected Properties
- @injected in razor

Demo

Mealsharing App

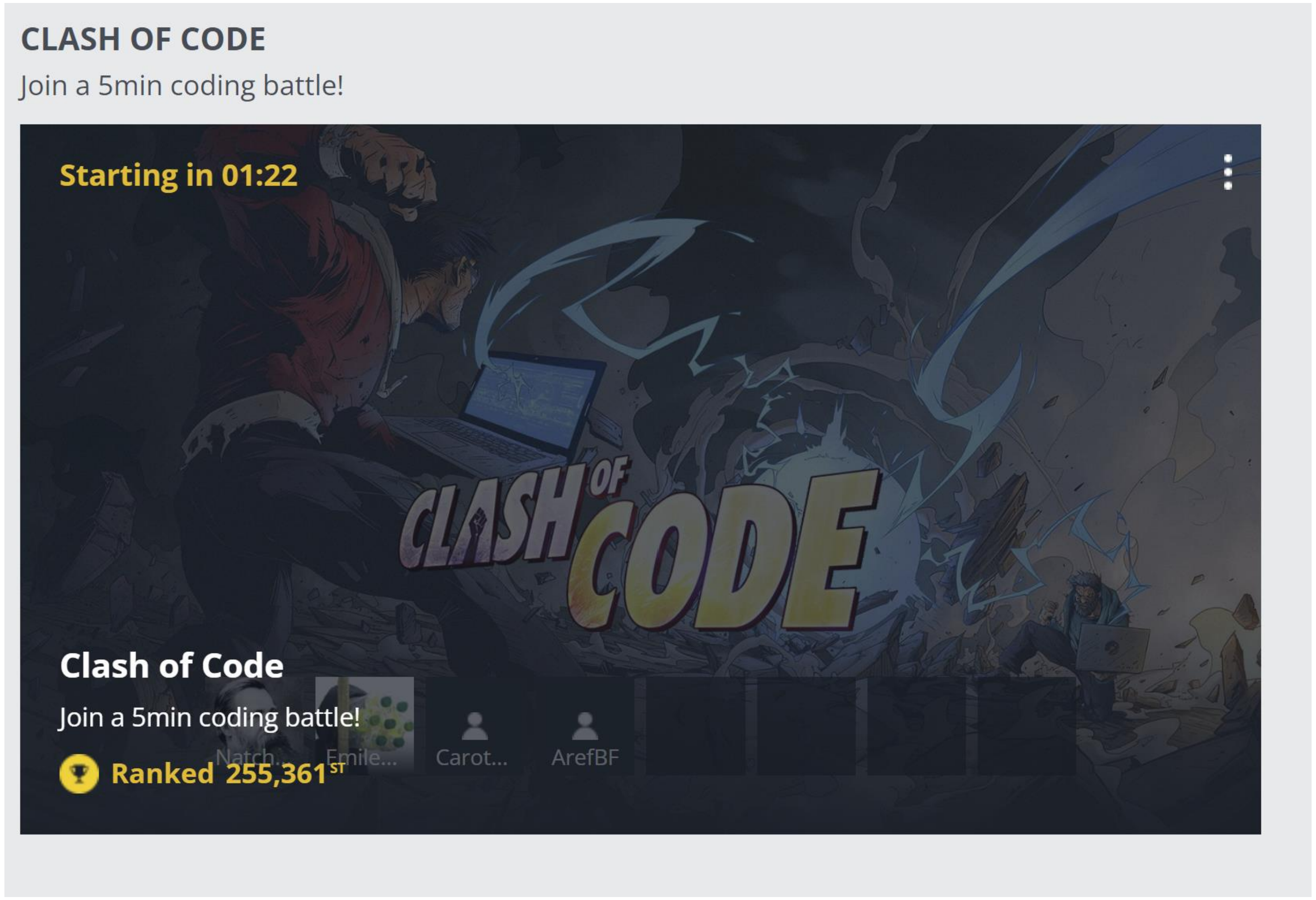
Web API

Which Models?

Which Controllers?

Which Services?

Codingame.com


CLASH OF CODE

Join a 5min coding battle!