Automapper vs Mapster

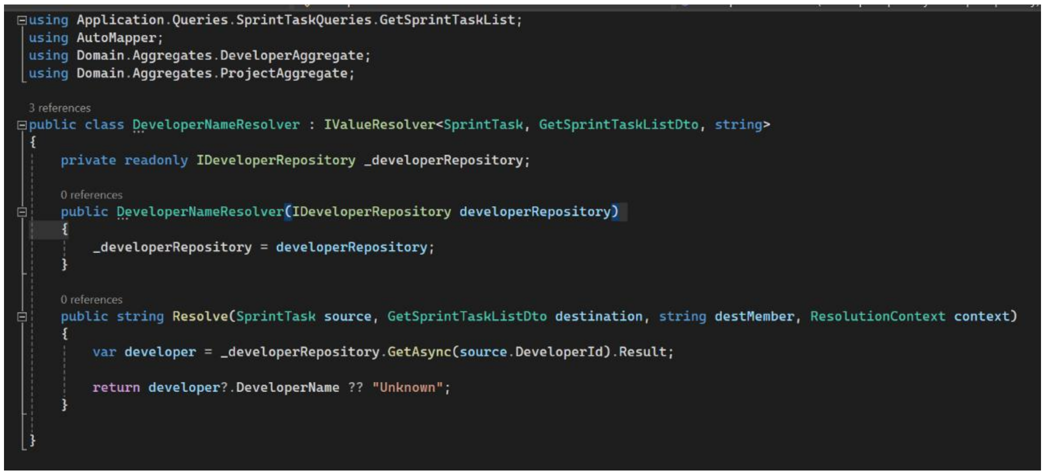
<u>automapper :</u>

- Convention Model:
  Follows the convention-over-configuration principle, automatically mapping properties with the same name.
- Fluent API: Uses a fluent API for configuration, allowing customization of mappings as needed.
- Feature-Rich: Comprehensive feature set supporting flattening, conditional mapping, nested mappings, and more.
- Ease of Use: Known for its ease of use, widely adopted, and features a large user base.
- Convention for Naming: Typically follows a naming convention where destination property names match source property names by default.

<u>mapster :</u>

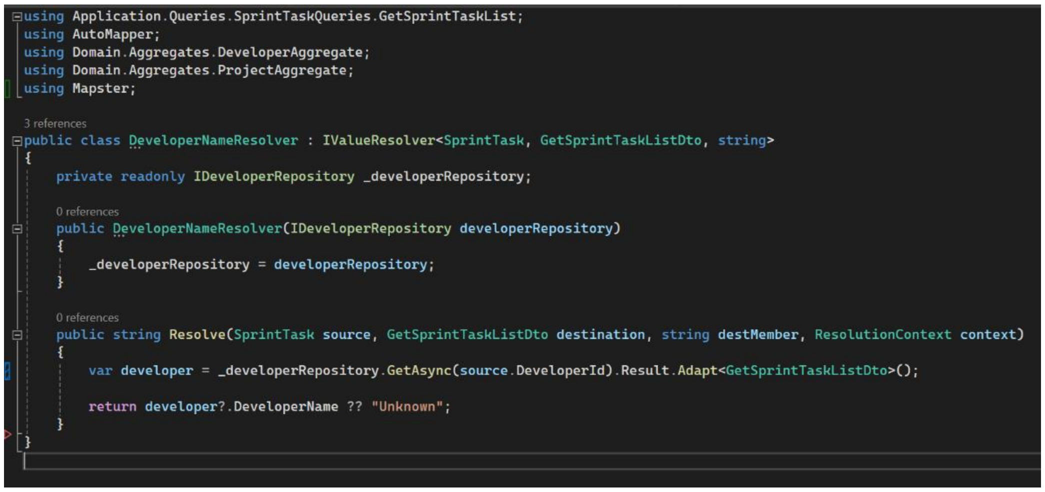
- Fluent API: Similar to AutoMapper, uses a fluent configuration API for mapping definitions.
- Performance: Emphasizes performance, claiming to be faster than AutoMapper in certain scenarios.
- Code Generation: Includes a code generation tool that can generate mapping code during compile time for enhanced performance.
- More Explicit: Considered more explicit, requiring more explicit mapping definitions.
- Immutable Objects: Good support for mapping to and from immutable objects.

Example to resolve developer name using automapper :



```
using Application.Queries.SprintTaskQueries.GetSprintTaskList;
using AutoMapper;
using Domain.Aggregates.DeveloperAggregate;
using Domain.Aggregates.ProjectAggregate;

3 references
public class DeveloperNameResolver : IValueResolver<SprintTask, GetSprintTaskListDto, string>
{
    private readonly IDeveloperRepository _developerRepository;

    0 references
    public DeveloperNameResolver(IDeveloperRepository developerRepository)
    {
        _developerRepository = developerRepository;
    }

    0 references
    public string Resolve(SprintTask source, GetSprintTaskListDto destination, string destMember, ResolutionContext context)
    {
        var developer = _developerRepository.GetAsync(source.DeveloperId).Result;

        return developer?.DeveloperName ?? "Unknown";
    }

}
```

Example to resolve developer name using mapster:



```
using Application.Queries.SprintTaskQueries.GetSprintTaskList;
using AutoMapper;
using Domain.Aggregates.DeveloperAggregate;
using Domain.Aggregates.ProjectAggregate;
using Mapster;

3 references
public class DeveloperNameResolver : IValueResolver<SprintTask, GetSprintTaskListDto, string>
{
    private readonly IDeveloperRepository _developerRepository;

    0 references
    public DeveloperNameResolver(IDeveloperRepository developerRepository)
    {
        _developerRepository = developerRepository;
    }

    0 references
    public string Resolve(SprintTask source, GetSprintTaskListDto destination, string destMember, ResolutionContext context)
    {
        var developer = _developerRepository.GetAsync(source.DeveloperId).Result.Adapt<GetSprintTaskListDto>();

        return developer?.DeveloperName ?? "Unknown";
    }
}
```

References:

- [MapsterMapper/Mapster: A fast, fun and stimulating object to object Mapper (github.com)]

- https://automapper.org/