Day 04. Reflection

# Чек-листы

## Общие

1. Выполнены ли общие требования?
2. Проверяемый может объяснить работу программы?
3. Названы ли переменные осмысленно и функционально?
4. Отсутствуют ли в коде хардкод и “магические числа”?
5. Соответствует ли код [C# Coding Conventions](https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/inside-a-program/coding-conventions)?
6. Программа запускается?
7. Соответствует ли структура проекта данной в задании?

## Задание 00

1. Подключен ли класс DefaultHttpContext с помощью nuget-пакета, а не скопирован в код?
2. Реализован ли вывод списка полей класса с помощью рефлексии и применяются ли при получении следующие флаги:

BindingFlags.Public |

BindingFlags.NonPublic |

BindingFlags.Instance |

BindingFlags.Static

1. Реализован ли вывод списка свойств класса с помощью рефлексии и применяются ли при получении следующие флаги:

BindingFlags.Public |

BindingFlags.Instance |

BindingFlags.Static

1. Реализован ли вывод списка методов класса с помощью рефлексии и применяются ли при получении следующие флаги:

BindingFlags.Public |

BindingFlags.Instance |

BindingFlags.Static

1. Возвращает ли приложение ожидаемый результат:

$ dotnet run

Type: Microsoft.AspNetCore.Http.DefaultHttpContext

Assembly: Microsoft.AspNetCore.Http, Version=2.2.2.0, Culture=neutral, PublicKeyToken=adb9793829ddae60

Based on: Microsoft.AspNetCore.Http.HttpContext

Fields:

Microsoft.AspNetCore.Http.Features.FeatureReferences`1[Microsoft.AspNetCore.Http.DefaultHttpContext+FeatureInterfaces] \_features

Microsoft.AspNetCore.Http.HttpRequest \_request

Microsoft.AspNetCore.Http.HttpResponse \_response

Microsoft.AspNetCore.Http.Authentication.AuthenticationManager \_authenticationManager

Microsoft.AspNetCore.Http.ConnectionInfo \_connection

Microsoft.AspNetCore.Http.WebSocketManager \_websockets

System.Func`2[Microsoft.AspNetCore.Http.Features.IFeatureCollection,Microsoft.AspNetCore.Http.Features.IItemsFeature] \_newItemsFeature

System.Func`2[Microsoft.AspNetCore.Http.Features.IFeatureCollection,Microsoft.AspNetCore.Http.Features.IServiceProvidersFeature] \_newServiceProvidersFeature

System.Func`2[Microsoft.AspNetCore.Http.Features.IFeatureCollection,Microsoft.AspNetCore.Http.Features.Authentication.IHttpAuthenticationFeature] \_newHttpAuthenticationFeature

System.Func`2[Microsoft.AspNetCore.Http.Features.IFeatureCollection,Microsoft.AspNetCore.Http.Features.IHttpRequestLifetimeFeature] \_newHttpRequestLifetimeFeature

System.Func`2[Microsoft.AspNetCore.Http.Features.IFeatureCollection,Microsoft.AspNetCore.Http.Features.ISessionFeature] \_newSessionFeature

System.Func`2[Microsoft.AspNetCore.Http.Features.IFeatureCollection,Microsoft.AspNetCore.Http.Features.ISessionFeature] \_nullSessionFeature

System.Func`2[Microsoft.AspNetCore.Http.Features.IFeatureCollection,Microsoft.AspNetCore.Http.Features.IHttpRequestIdentifierFeature] \_newHttpRequestIdentifierFeature

Properties:

Microsoft.AspNetCore.Http.Features.IFeatureCollection Features

Microsoft.AspNetCore.Http.HttpRequest Request

Microsoft.AspNetCore.Http.HttpResponse Response

Microsoft.AspNetCore.Http.ConnectionInfo Connection

Microsoft.AspNetCore.Http.Authentication.AuthenticationManager Authentication

Microsoft.AspNetCore.Http.WebSocketManager WebSockets

System.Security.Claims.ClaimsPrincipal User

System.Collections.Generic.IDictionary`2[System.Object,System.Object] Items

System.IServiceProvider RequestServices

System.Threading.CancellationToken RequestAborted

System.String TraceIdentifier

Microsoft.AspNetCore.Http.ISession Session

Methods:

IFeatureCollection get\_Features ()

HttpRequest get\_Request ()

HttpResponse get\_Response ()

ConnectionInfo get\_Connection ()

AuthenticationManager get\_Authentication ()

WebSocketManager get\_WebSockets ()

ClaimsPrincipal get\_User ()

Void set\_User (ClaimsPrincipal value)

IDictionary`2 get\_Items ()

Void set\_Items (IDictionary`2 value)

IServiceProvider get\_RequestServices ()

Void set\_RequestServices (IServiceProvider value)

CancellationToken get\_RequestAborted ()

Void set\_RequestAborted (CancellationToken value)

String get\_TraceIdentifier ()

Void set\_TraceIdentifier (String value)

ISession get\_Session ()

Void set\_Session (ISession value)

Void Initialize (IFeatureCollection features)

Void Uninitialize ()

Void Abort ()

Type GetType ()

String ToString ()

Boolean Equals (Object obj)

Int32 GetHashCode ()

## Задание 01

1. Подключен ли класс DefaultHttpContext с помощью nuget-пакета, а не скопирован в код?
2. Реализовано ли получение информации о свойстве и заполнение его через рефлексию?
3. Соответствует ли вывод следующему:

$ dotnet run

Old Response value: Microsoft.AspNetCore.Http.Internal.DefaultHttpResponse

New Response value:

## Задание 02

1. Является ли метод SetValues класса ConsoleSetter обобщенным, соответствует ли его сигнатрура: SetValues<T>(T input)? Ограничен ли тип T конструкцией where T: class?
2. Выводятся ли имя типа для приветствия с помощью рефлексии, не захардкожено ли оно?
3. Реализован ли атрибут NoDisplayAttribute с наследованием System.Attribute и ограничениями использования: AttributeTargets.Property, AllowMultiple = false?
4. Отмечены ли свойства UserName, Email и PhoneNumber атрибутами Description и DefaultValue, а все прочие свойства атрибутом NoDisplay?
5. Используется ли в реализации метода SetValues фильтрация по NoDisplay, вывод по Description и запись по DefaultValue?
6. Соответствует ли результат работы приложения следующему для IdentityUser:

$ dotnet run

Let's set IdentityUser!

Set User name:

Ann

Set Email address:

ann@test.ru

Set Phone number:

123000123

We've set our instance!

User: Ann, ann@test.ru, 123000123

1. Соответствует ли результат работы приложения следующему для IdentityUser:

$ dotnet run

Let's set IdentityUser!

Set User name:

Set Email address:

Set Phone number:

We've set our instance!

User: Me, test@test, 1234567890

1. Соответствует ли результат работы приложения следующему для IdentityRole:

$ dotnet run

Let's set IdentityRole!

Set Name:

Moderator

Set Description:

A role for moderation

We've set our instance!

Moderator, A role for moderation

## Задание 03

1. Является ли методы класса TypeFactory обобщенными? Ограничен ли тип T для них конструкцией where T: class?
2. Реализует ли метод CreateWithConstructor создание объекта с помощью рефлексии и метода GetConstructor?
3. Реализует ли метод CreateWithActivator создание объекта с помощью рефлексии и класса Activator?
4. Нет ли в классе TypeFactory зависимостей от классов IdentityUser и IdentityRole?
5. Выводит ли приложение в консоль:

d04\_ex03.Models.IdentityUser

user1 != user2

d04\_ex03.Models.IdentityRole

role1 != role2

1. Может ли проверяемый объяснить, почему объекты, получаемые в результате работы этих методов, не равны?
2. Бонус: Является ли метод CreateWithParameters обобщенным и реализует ли он создание объекта с помощью рефлексии с учетом входных параметров?
3. Бонус: Соответствует ли вывод в консоль следующему:

$ dotnet run

d04\_ex03.Models.IdentityUser

user1 != user2

d04\_ex03.Models.IdentityRole

role1 != role2

d04\_ex03.Models.IdentityUser

Set name:

Activated user

Username set: Activated user

1. Бонус: Соответствует ли вывод в консоль следующему:

$ dotnet run

d04\_ex03.Models.IdentityUser

user1 != user2

d04\_ex03.Models.IdentityRole

role1 != role2

d04\_ex03.Models.IdentityUser

Set name:

My new name

Username set: My new name