ASP.NET Core. Веб-приложения

Декомпозиция слоев приложения



# На этом уроке

1. Освоим навыки декомпозиции слоев в шаблоне MVC
2. Рассмотрим принципы SOLID & DRY и применим их на практике
3. Декомпозируем слои приложения в Timesheets
4. Реализуем бизнес-логику приложения Timesheets

# 

Оглавление

[На этом уроке](#_4wfp8dsrmxhk)

[Знакомство с шаблоном MVC](#_5apa97np2url)

[Преимущества использования MVC](#_exk8zuz18anw)

[ASP.NET Core MVC](#_pas9504050yl)

[SOLID](#_w0ntjqrvv3vo)

[Single Responsibility](#_i2m5mwoc8g2a)

[Open-Closed](#_5ky1wsfky6rp)

[Liskov Substitution](#_ern766tbrvp)

[Interface Segregation](#_pmm3h74lfivc)

[Dependency Inversion](#_u9083gig7qv9)

[Глоссарий](#_tyjcwt)

[Дополнительные материалы](#_44sinio)

[Используемые источники](#_z337ya)

# 

# 

# 

# 

# 

# 

# 

# Знакомство с шаблоном MVC

Термин MVC (Model-View-Controller) впервые был использован еще в конце 1970-х, когда разработчики стали задумываться об организации кода в приложениях с графической составляющей (GUI).

Основная концепция предполагает разделение кодовой базы на три составляющих:

* Модель — это слой, содержащий данные, с которыми взаимодействуют пользователи приложения. Модели могут включать в себя как информацию, которая должна быть передана от слоя к слою, так и бизнес-логику по работе с этими данными. В таком случае они еще называются доменными моделями (domain models).
* Представление — это слой визуальной части приложения, которая предоставляет пользователю графический интерфейс взаимодействия.
* Контроллер — это слой, связывающий пользовательские действия с бизнес-логикой приложения. Иными словами, это уровень обработки входящих запросов с дальнейшей передачей намерений в слой бизнес-логики и хранения данных.

Таким образом, взаимодействия в шаблоне MVC можно изобразить так:



Пользовательский запрос попадает в контроллер, затем направляется в нужную модель, которая заполняется данными из хранилища и возвращается обратно в виде ответа на представление, где и происходит рендеринг полученной информации.

Благодаря шаблону MVC реализуется важная концепция разделения ответственности (separation of concerns). Так, например, логика представления совершенно не связана с доменом приложения. Между слоями есть лишь контракты взаимодействия: представление ожидает в качестве ответа модель определенного рода. То, каким образом эта модель данных будет подготовлена, уже вне зоны ответственности представления. И наоборот: домен приложения обязуется выполнить заложенную логику, но то, что произойдет на интерфейсной части, для него совершенно не важно.

# Преимущества использования MVC

Продолжая разговор о принципе разделения ответственностей, мы рассмотрим преимущества использования шаблона MVC, чтобы понять, почему этот архитектурный стиль стал так популярен и востребован в современной разработке.

Во-первых, мы получаем облегченную масштабируемость приложения. Стоит отметить, что речь идет о масштабируемости логики, а не производительности. Разделение ответственности по независимым друг от друга слоям позволяет достичь слабой связности кода между конкретными компонентами.

Во-вторых, мы можем внедрять unit-тестирование отдельных слоев приложения.

В-третьих, такой шаблон дает гибкость. Это частично связано и с масштабируемостью. Слабая связность компонентов положительно сказывается на возможности доработки каждого из них по отдельности, не опираясь на зависимости и требования других слоев.

# ASP.NET Core MVC

Фреймворк ASP.NET Core представляет собой платформу, которая максимально адаптирована к использованию шаблона MVC. А ещё этот фреймворк содержит множество полезных компонентов, которые доступны «из коробки»:

* Контроллеры
* Настройка маршрутизации запросов
* Привязки и проверки моделей данных
* Фильтры
* Внедрение зависимостей

Более подробно ознакомиться с функционалом фреймворка можно на официальном сайте, где представлена исчерпывающая документация. Ссылки на неё находятся в разделе «дополнительные материалы».

# SOLID

Принципы SOLID являются неотъемлемой частью разработки в объектно-ориентированной парадигме. Само по себе SOLID является акронимом следующих понятий:

* Single Responsibility (принцип единственной ответственности)
* Open-Closed (принцип открытости-закрытости)
* Liskov Substitution (принцип подстановки Барбары Лисков)
* Interface Segregation (принцип разделения интерфейсов)
* Dependency Inversion (принцип инверсии зависимостей)

Следование правилам SOLID помогает сделать код более читаемым и гранулированным. Как следствие, упрощается масштабирование и поддержка кода. Теперь рассмотрим каждое из понятий подробнее.

## Single Responsibility

Класс должен отвечать только за одну бизнес-ответственность.

Представим, что нам нужно добавить отчет в виде csv-файла по транзакциям пользователя за выбранный период. Чтобы сгенерировать его, нам необходимо сначала выбрать данные из хранилища, затем преобразовать их и сохранить в виде csv-таблицы. В рамках подготовки отчета мы выделили три бизнес-ответственности, которые никак не зависят друг от друга. И действительно, слой выбора данных из хранилища не должен знать о том, где и как будет использоваться полученная информация. Возможно, эти данные будут использоваться не только для отчета, а например, для отправки в иное хранилище. При этом слой обработки информации не должен знать о том, куда будут передаваться готовые данные отчетности: это может быть сохранение в csv-файл, pdf-файл или же отправка e-mail.

Так, если мы реализуем функционал по выборке, преобразованию и сохранению данных файлов в рамках одного класса, сложно будет организовать дальнейшую поддержку, когда возникнет необходимость добавить новый формат отчета.

## Open-Closed

Классы должны быть открыты для расширения, но закрыты для модификации.

## Liskov Substitution

Принцип подстановки Барбары Лисков заключается в том, чтобы классы-наследники могли служить заменой для родительских классов, не нарушая заложенную логику.

## Interface Segregation

Правило говорит о разделение интерфейсов. Классы, реализующие интерфейсы, не должны зависеть от методов, которые они не используют.

Иными словами, необходимо создавать узкоспециализированные интерфейсы.

## Dependency Inversion

Принцип инверсии зависимостей заключается в том, что модули верхних уровней не должны зависеть от модулей нижних уровней. При этом все модули должны зависеть от абстракций. А абстракции, в свою очередь, не должны зависеть от деталей реализации.

# Практическое задание

Необходимо реализовать ASP.NET Core MVC приложение с разделением логики на уровень контроллеров, сервисов и репозиториев. Для уровня сервисов и репозиториев нужно использовать интерфейсы. Приветствуется соблюдение принципов SOLID а также применение Dependency Injection встроенными средствами платформы ASP.NET Core.

Предлагаем создать api для работы с коллекцией persons:

* GET /persons/{id} — получение человека по идентификатору
* GET /persons/search?searchTerm={term} — поиск человека по имени
* GET /persons/?skip={5}&take={10} — получение списка людей с пагинацией
* POST /persons — добавление новой персоны в коллекцию
* PUT /persons — обновление существующей персоны в коллекции
* DELETE /persons/{id} — удаление персоны из коллекции

Класс Person:

|  |
| --- |
| public class Person {  public int Id { get; set; }  public string FirstName { get; set; }  public string LastName { get; set; }  public string Email { get; set; }  public string Company { get; set; }  public int Age { get; set; } } |

Набор сгенерированных данных (для репозитория):

|  |
| --- |
| var data = new List<Person>() { new Person { Id = 1, FirstName = "Veda", LastName = "Richmond", Email = "ligula@necluctus.edu", Company = "Quisque Ac Libero LLP", Age = 42 }, new Person { Id = 2, FirstName = "Demetria", LastName = "Andrews", Email = "feugiat.metus@penatibuset.org", Company = "Nulla Facilisi Foundation", Age = 31 }, new Person { Id = 3, FirstName = "Byron", LastName = "Holmes", Email = "neque.Sed.eget@non.co.uk", Company = "Et Associates", Age = 63 }, new Person { Id = 4, FirstName = "Alexander", LastName = "Cummings", Email = "egestas.ligula@ultricesDuisvolutpat.ca", Company = "Vel Institute", Age = 23 }, new Person { Id = 5, FirstName = "Melinda", LastName = "Miles", Email = "justo.nec.ante@nonummyFusce.ca", Company = "Eu Nibh Vulputate Company", Age = 64 }, new Person { Id = 6, FirstName = "Dustin", LastName = "Beck", Email = "iaculis@afeugiat.edu", Company = "Nec Diam Incorporated", Age = 35 }, new Person { Id = 7, FirstName = "Ralph", LastName = "Maddox", Email = "ipsum@vulputatelacus.co.uk", Company = "Enim Corp.", Age = 22 }, new Person { Id = 8, FirstName = "Levi", LastName = "Zamora", Email = "lectus.a.sollicitudin@nuncQuisque.com", Company = "Sodales At Velit Corp.", Age = 57 }, new Person { Id = 9, FirstName = "Driscoll", LastName = "Estrada", Email = "Phasellus@Craspellentesque.org", Company = "Id Mollis Nec LLC", Age = 37 }, new Person { Id = 10, FirstName = "Hiram", LastName = "Mejia", Email = "lacus.Mauris@semper.ca", Company = "Donec Tincidunt Donec Industries", Age = 59 }, new Person { Id = 11, FirstName = "Mason", LastName = "Jefferson", Email = "Integer.vitae.nibh@nibh.co.uk", Company = "Lectus Justo Ltd", Age = 65 }, new Person { Id = 12, FirstName = "Nigel", LastName = "Rich", Email = "id@faucibusleoin.net", Company = "Tristique Ac Ltd", Age = 52 }, new Person { Id = 13, FirstName = "Tarik", LastName = "Hughes", Email = "enim@ultricesDuisvolutpat.edu", Company = "Lacus Varius Et Associates", Age = 58 }, new Person { Id = 14, FirstName = "Wallace", LastName = "Gross", Email = "Curabitur.ut.odio@anteMaecenasmi.co.uk", Company = "Rhoncus Id Corporation", Age = 29 }, new Person { Id = 15, FirstName = "Arden", LastName = "Rivers", Email = "magna.nec.quam@lobortis.net", Company = "Vivamus Corporation", Age = 59 }, new Person { Id = 16, FirstName = "Vincent", LastName = "Fox", Email = "faucibus.Morbi.vehicula@ipsumdolor.edu", Company = "Imperdiet Dictum Magna Foundation", Age = 54 }, new Person { Id = 17, FirstName = "Aphrodite", LastName = "Randolph", Email = "ac@scelerisquesedsapien.org", Company = "Mattis Foundation", Age = 27 }, new Person { Id = 18, FirstName = "Alisa", LastName = "Riggs", Email = "montes@scelerisque.edu", Company = "Rutrum Non Hendrerit Consulting", Age = 25 }, new Person { Id = 19, FirstName = "Jaime", LastName = "Lott", Email = "velit.Quisque.varius@aliquetmolestie.net", Company = "Ut LLC", Age = 61 }, new Person { Id = 20, FirstName = "Jamalia", LastName = "Buchanan", Email = "arcu.eu.odio@congue.ca", Company = "Curabitur Sed Tortor Ltd", Age = 61 }, new Person { Id = 21, FirstName = "Raya", LastName = "Mckenzie", Email = "Integer.sem.elit@bibendumsedest.net", Company = "In Inc.", Age = 43 }, new Person { Id = 22, FirstName = "Dante", LastName = "Blackwell", Email = "Cras.eget.nisi@Vestibulum.edu", Company = "Nec Foundation", Age = 48 }, new Person { Id = 23, FirstName = "Kato", LastName = "Dickson", Email = "facilisis@doloregestas.co.uk", Company = "Augue Scelerisque Institute", Age = 60 }, new Person { Id = 24, FirstName = "Clio", LastName = "Shaffer", Email = "tincidunt@eget.edu", Company = "Dui Augue Eu Limited", Age = 29 }, new Person { Id = 25, FirstName = "Hamilton", LastName = "Kidd", Email = "magna@felisegetvarius.net", Company = "Enim Incorporated", Age = 26 }, new Person { Id = 26, FirstName = "Kerry", LastName = "Oneil", Email = "Suspendisse.eleifend@Crasdolor.com", Company = "Interdum Inc.", Age = 48 }, new Person { Id = 27, FirstName = "Mohammad", LastName = "Thompson", Email = "elit.pretium.et@malesuadafamesac.com", Company = "Facilisis Eget Ipsum Inc.", Age = 34 }, new Person { Id = 28, FirstName = "Vernon", LastName = "Cardenas", Email = "felis@conguea.org", Company = "Iaculis Quis Consulting", Age = 35 }, new Person { Id = 29, FirstName = "Murphy", LastName = "Weaver", Email = "Proin@feugiatnecdiam.org", Company = "Integer Urna Institute", Age = 63 }, new Person { Id = 30, FirstName = "Xena", LastName = "Hart", Email = "facilisis.facilisis.magna@loremutaliquam.net", Company = "Orci Industries", Age = 39 }, new Person { Id = 31, FirstName = "Ivor", LastName = "Lara", Email = "Proin.ultrices.Duis@lacuspede.com", Company = "Ante Foundation", Age = 30 }, new Person { Id = 32, FirstName = "Dana", LastName = "Merritt", Email = "et.magnis@Sed.edu", Company = "Eget Industries", Age = 53 }, new Person { Id = 33, FirstName = "Brielle", LastName = "Woodward", Email = "elit.Nulla@magna.edu", Company = "Lorem Vehicula Et Foundation", Age = 45 }, new Person { Id = 34, FirstName = "Hasad", LastName = "Duran", Email = "et@nislsem.co.uk", Company = "Magna Suspendisse Consulting", Age = 49 }, new Person { Id = 35, FirstName = "Quamar", LastName = "Moses", Email = "Proin.sed.turpis@imperdiet.co.uk", Company = "Eros Institute", Age = 32 }, new Person { Id = 36, FirstName = "Scarlet", LastName = "Barlow", Email = "nisl.sem.consequat@idnunc.co.uk", Company = "Aenean Massa Consulting", Age = 58 }, new Person { Id = 37, FirstName = "Courtney", LastName = "Foley", Email = "urna@mauris.org", Company = "Mauris Associates", Age = 47 }, new Person { Id = 38, FirstName = "Kennedy", LastName = "Shields", Email = "Cras@Nullam.org", Company = "Id Nunc Interdum LLC", Age = 40 }, new Person { Id = 39, FirstName = "Eve", LastName = "Maynard", Email = "metus.sit@lorem.ca", Company = "Pellentesque Ultricies Associates", Age = 30 }, new Person { Id = 40, FirstName = "Debra", LastName = "Ellis", Email = "Nullam@pretium.ca", Company = "Nulla Tincidunt Industries", Age = 24 }, new Person { Id = 41, FirstName = "Vivian", LastName = "Mcguire", Email = "ornare@at.net", Company = "Id Consulting", Age = 48 }, new Person { Id = 42, FirstName = "Jason", LastName = "Mckinney", Email = "tempor.augue@purusNullam.com", Company = "Netus Et Inc.", Age = 48 }, new Person { Id = 43, FirstName = "Patrick", LastName = "Small", Email = "fringilla@Proinsed.co.uk", Company = "Hendrerit Institute", Age = 61 }, new Person { Id = 44, FirstName = "Drew", LastName = "Travis", Email = "scelerisque.scelerisque@velit.org", Company = "Penatibus Corp.", Age = 55 }, new Person { Id = 45, FirstName = "Burke", LastName = "Miller", Email = "Suspendisse@aliquet.net", Company = "Quis Diam Pellentesque PC", Age = 41 }, new Person { Id = 46, FirstName = "Ralph", LastName = "Medina", Email = "Class.aptent.taciti@mauris.edu", Company = "Lorem Ipsum Dolor Corp.", Age = 55 }, new Person { Id = 47, FirstName = "Alana", LastName = "Madden", Email = "at.velit.Cras@aptenttacitisociosqu.net", Company = "Euismod Est Arcu Institute", Age = 33 }, new Person { Id = 48, FirstName = "Salvador", LastName = "Cohen", Email = "magna.Duis@Phasellus.org", Company = "Purus PC", Age = 37 }, new Person { Id = 49, FirstName = "Jenette", LastName = "Dejesus", Email = "adipiscing.Mauris.molestie@liberoduinec.ca", Company = "Lectus Justo Incorporated", Age = 56 }, new Person { Id = 50, FirstName = "Ramona", LastName = "Gilliam", Email = "massa.Vestibulum@lectuspede.ca", Company = "Imperdiet Dictum LLP", Age = 24 },  }; |

# Дополнительные материалы

1. Pro ASP.NET Core MVC 2, Adam Freeman
2. <https://docs.microsoft.com/ru-ru/aspnet/core/mvc/overview?view=aspnetcore-5.0#web-apis>
3. Ссылка на репозиторий проекта Timesheets: <https://github.com/evgshk/gb-ts-app>

# Используемые источники

1. <https://docs.microsoft.com/ru-ru/aspnet/core/mvc/overview?view=aspnetcore-5.0#web-apis>
2. <https://ru.wikipedia.org/wiki/SOLID_(%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)>