Инет комопненты клиент сервра типы взиам веб приол http сосотояние сеския https рукопожание

Способы сохранить состояние(куки, сессия)

Как работает какие есть спосбы передать ок компонетам запрос можно ли вперепр назад

**Объяснение Middleware:**

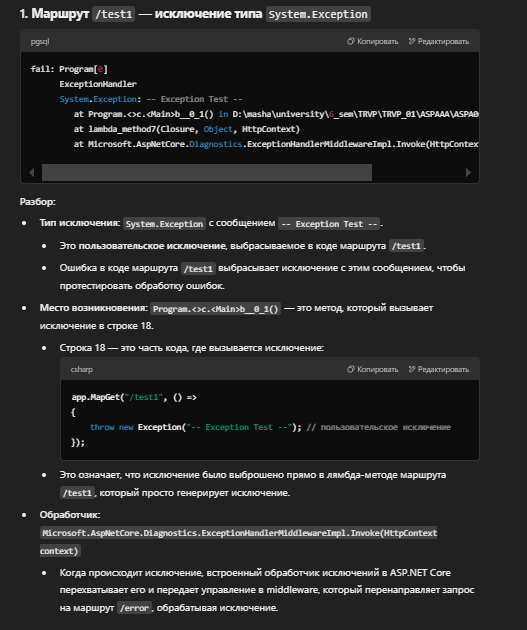
* Middleware — это компонент, который обрабатывает HTTP-запросы и ответы на уровне приложения, промежуточный слой между клиентом и сервером. Он может выполнять различные действия, такие как аутентификация, логирование, обработка ошибок, управление сессиями, маршрутизация запросов и многое другоеВ данном случае UseWelcomePage перехватывает запросы на /aspnetcore и возвращает готовую приветственную страницу.

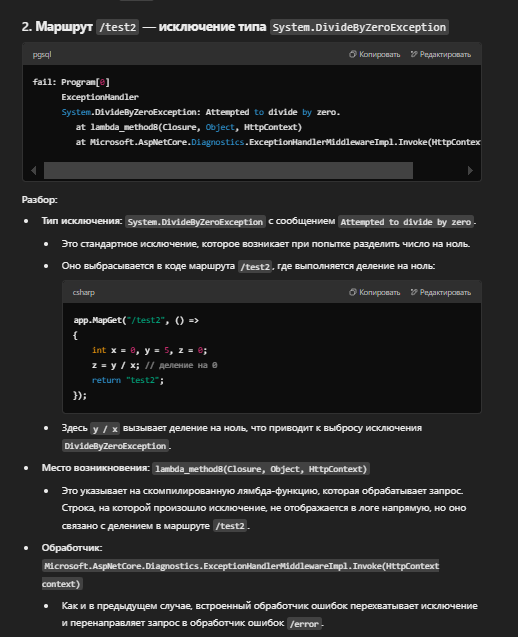
**Поясните проблему с обработчиком MapGet:**

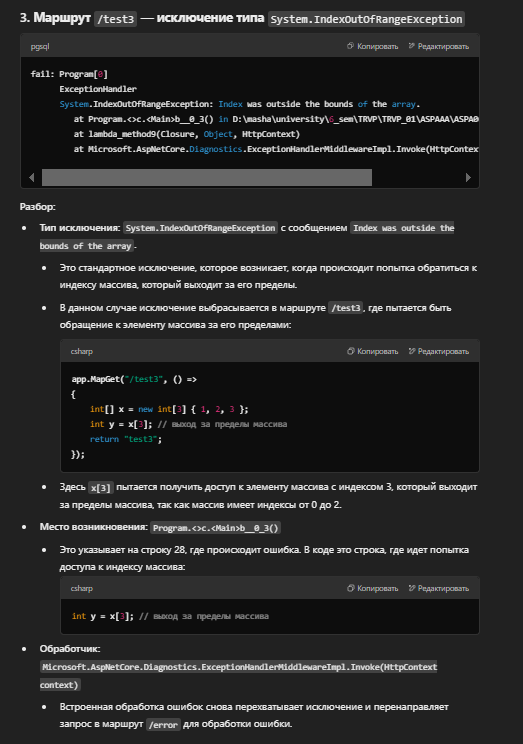
* Если после UseWelcomePage добавлен MapGet("/aspnetcore", ...), он не будет работать, так как UseWelcomePage уже перехватил запрос и завершил его обработку.
* В вашем приложении при возникновении исключений в одном из маршрутов (/test1, /test2, /test3), они передаются в стандартный обработчик ошибок UseExceptionHandler("/error").
* Когда приложение перехватывает исключение, оно записывает это исключение в журнал с помощью logger.LogError(), в том числе с информацией о типе исключения, его сообщении и месте в коде.
* Ошибка логируется с указанием пути и строки, где произошла ошибка, что помогает разработчику понять, в каком месте в коде она возникла.

**Примечание:**

Этот вывод работает корректно в **Release** режиме, так как в **Debug** режиме могут быть дополнительно выведены другие подробности, такие как стек вызовов в браузере, что небезопасно для продакшн-окружения.







1. **Поясните назначение паттерна Builder. Укажите: в каком месте ASPA применяется этот паттерн.**

Паттерн Builder используется для создания сложных объектов поэтапно. Он позволяет разделить процесс создания объекта от его представления, что особенно полезно, когда объект имеет много параметров или конфигураций.

**Применение в ASP.NET Core**: В ASP.NET Core паттерн Builder применяется в классе WebApplicationBuilder, который используется для конфигурирования приложения, включая настройки сервисов и middleware.

В ASPA паттерн Builder часто используется в создании сложных HTTP-ответов или запросов. Например, в построении сложных конфигураций middleware или в формировании сложных объектов для передачи данных (например, в случае API-ответов с различными параметрами).

1. **Поясните назначение паттерна Chain of Responsibility. Укажите: в каком месте ASPA применяется этот паттерн.**

Паттерн Chain of Responsibility позволяет передавать запрос по цепочке обработчиков, где каждый обработчик может обработать запрос или передать его дальше. Это позволяет избежать жесткой привязки между отправителем и получателем запроса.

Этот паттерн используется в ASPA для обработки HTTP-запросов в цепочке middleware. Каждое middleware-компонент проверяет запрос и решает, как его обработать (например, проверить аутентификацию или обработать ошибку). Запрос проходит через цепочку, пока не будет обработан или не дойдет до конца.

1. **Поясните понятие: Middleware.**

Middleware — это компонент, который обрабатывает HTTP-запросы и ответы на уровне приложения, промежуточный слой между клиентом и сервером. Он может выполнять различные действия, такие как аутентификация, логирование, обработка ошибок, управление сессиями, маршрутизация запросов и многое другое.

1. **Поясните понятие: статический файл.**

Статический файл — это файл, который не изменяется при запросе (например, изображения, стили, JavaScript-файлы). Эти файлы отправляются клиенту как есть, без каких-либо изменений со стороны сервера.

1. **Какой отличительный (общепринятый) признак наименования, подключаемого Middleware-компонента.**

Общепринятый признак наименования подключаемого middleware-компонента — это использование метода app.UseXXXXX(), где XXXXX указывает на функциональность middleware, например, app.UseRouting() или app.UseAuthentication().

1. **Перечислите известные вам встроенные Middleware-компоненты.**

****  UseRouting — для маршрутизации запросов.

 UseAuthentication — для аутентификации пользователей.

 UseAuthorization — для авторизации пользователей.

 UseStaticFiles — для обработки статических файлов.

 UseSession — для управления сессиями.

 UseCors — для управления политикой CORS (кросс-доменные запросы).

 UseExceptionHandler — для обработки исключений.

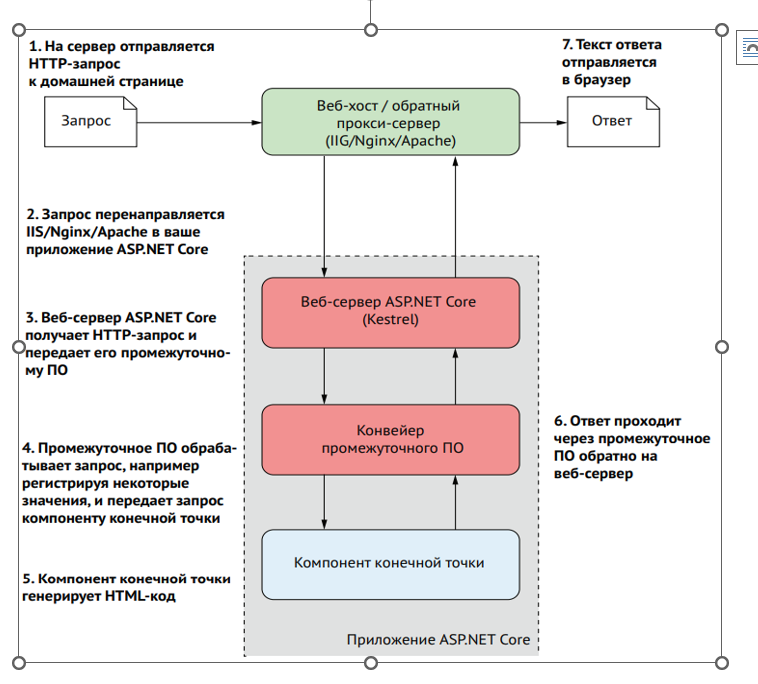
 UseEndpoints — для обработки конечных точек маршрутизации.

1. **Какое имя директория ASPA (по умолчанию), применяемого для хранения статических файлов приложения.**

По умолчанию для хранения статических файлов используется директория wwwroot. Она предназначена для размещения файлов, доступных для прямых HTTP-запросов, таких как изображения, стили и скрипты.

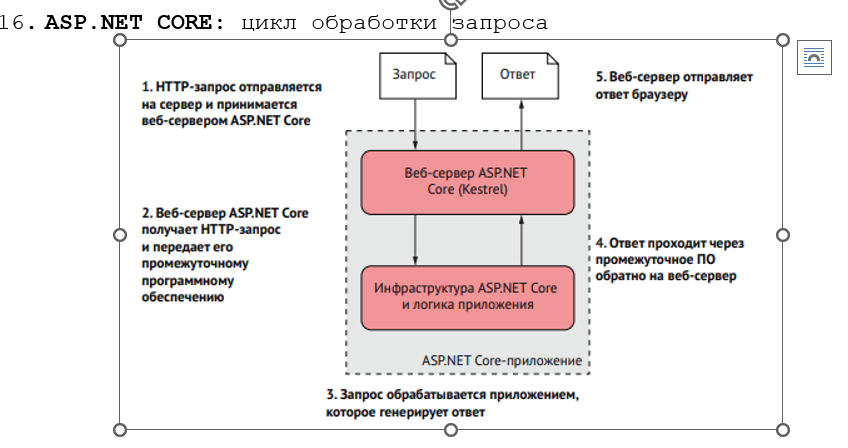
1. **♣ Кто такой Джон фон Нейман?☺**

Джон фон Нейман был выдающимся математиком, физиком и компьютерным ученым. Он внёс ключевой вклад в развитие теории игры, квантовой механики, а также в создание концепции архитектуры компьютера, которая стала основой современных вычислительных машин — архитектуры фон Неймана. В её основе лежит идея использования одной памяти для хранения как данных, так и инструкций.



Как работает какие есть спосбы передать ок компонетам запрос можно ли вперепр назад

1. **ASP.NET CORE:** программная платформа для разработки web-приложений**.**
2. **ASP.NET CORE:** кроссплатформенный фреймворк (Windows, Linux, Mac OS) с открытым кодом для разработки web-приложений на платформе .NET.



**HttpContext** – объект, создаваемый HTTP-сервером, содержит исчерпывающую информацию о соединении, запросе, будущем ответе.

1. **ASP.NET CORE:** этапы подготовки и запуска приложения:

- **generate** – создание приложения на основе шаблона;

- **restore** – восстановление пакеты и зависимости;

- **build** – компиляция и генерация ресурсов;

- **run**  - запуск скомпилированного приложения.

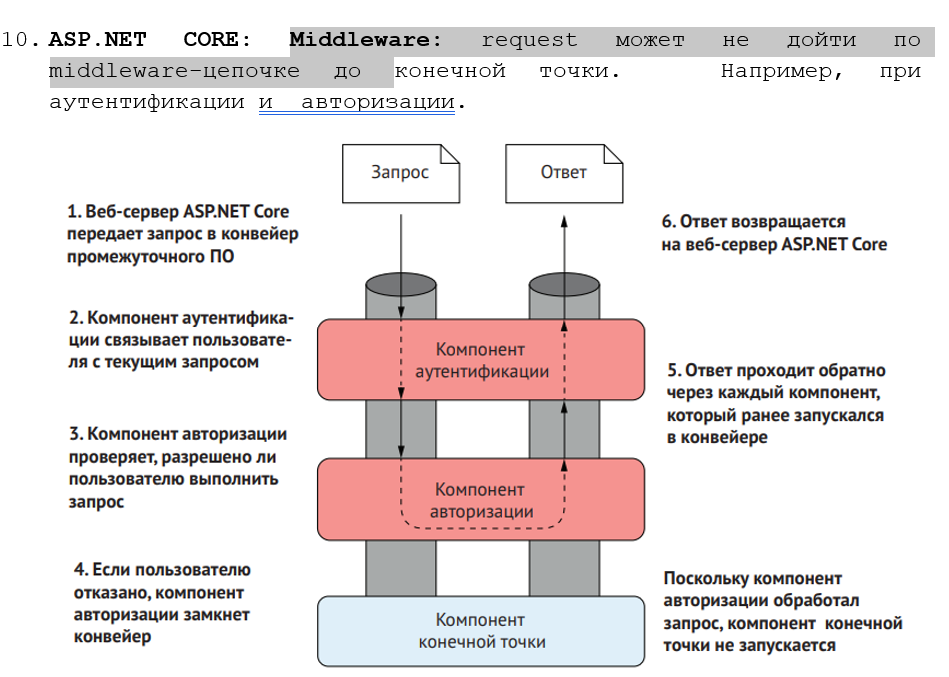
1. **Middleware:** двунаправленный конвейер обработки запросов (прямое направление) и ответов (обратное направление).
2. **Middleware:** Middleware**-**объект может одно из:

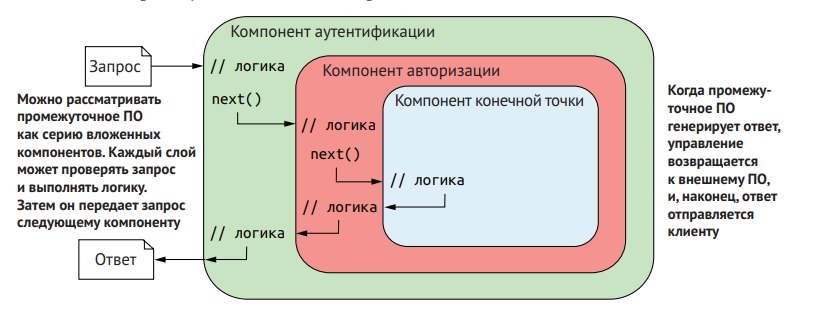
*на прямом пути*

1. получить request, сгенерировать response и передать его предшествующему Middleware**-**объекту или http-серверу;
2. получить request,изменить request и передать его следующему Middleware**-**объекту;

*на обратном пути*

получить исходящий response, изменить response, передать его следующему Middleware**-**объекту или http-серверу;



1. элемент, замыкающий конвейер и возвращающий ответ, называется **терминальным Middleware- элементом.**
2. **конвейер** - последовательность вложенных вызовов Middleware**-**элементов.
3. ****

Компоненты

1. Запрос:
   * Это входящий HTTP-запрос от клиента, который проходит через цепочку middleware.
2. Компонент аутентификации:
   * Отвечает за проверку личности пользователя.
   * Логика внутри этого компонента может включать проверку токена или учетных данных.
   * Если аутентификация успешна, запрос передается дальше с помощью next().
3. Компонент авторизации:
   * Отвечает за проверку прав пользователя на доступ к определенному ресурсу.
   * Логика может включать проверку ролей или прав доступа.
   * Если авторизация успешна, запрос передается дальше с помощью next().
4. Компонент конечной точки:
   * Это финальная часть цепочки, которая обрабатывает запрос и генерирует ответ.
   * Здесь выполняется основная бизнес-логика, например, получение данных из базы данных.

Ответ

* Когда промежуточное ПО генерирует ответ, он возвращается через ту же цепочку, но в обратном направлении:
  + Ответ может быть изменен на каждом уровне, например, добавлением заголовков или изменением формата данных.
  + Наконец, ответ отправляется обратно клиенту.

Логика работы

1. Обработка запроса:
   * Запрос проходит через компонент аутентификации, где проверяются данные пользователя.
   * Затем он проходит через компонент авторизации, где проверяются права доступа.
   * Если все проверки успешны, запрос передается к компоненту конечной точки.
2. Обработка ответа:
   * Ответ, сгенерированный компонентом конечной точки, проходит обратно через компоненты авторизации и аутентификации.
   * На каждом этапе ответ может быть изменен или дополнен.
   * В конечном итоге, ответ отправляется клиенту.

