Сучасні інтернет технології. Лабораторне заняття №3
КОНФІГУРАЦІЯ ПРОЄКТУ ЗАСТОСУНКУ ASP.NET CORE

Завдання для виконання (max - 100 балів)
1. Забезпечте проєкт файлами sharedsettings.json, appsettings.Development.json та appsettings.Production.json. Налаштуйте різні значення параметра ApplicationName та принаймні одного специфічного для свого проєкту параметра для кожного середовища. (max - 15 балів)

appsettings.Development.json:

```
{
  "ApplicationName": "MyApp (Development)",
  "CustomSettings": {
    "Message": "Welcome from DEV environment!"
  }
}
```

sharedsettings.json:

```
{
  "ApplicationName": "MyApp Shared",
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  }
}
```

appsettings.Production.json:

```
{
  "ApplicationName": "MyApp (Production)",
  "CustomSettings": {
    "Message": "Welcome from PROD environment!"
  }
}
```

Налаштування порядку завантаження:

```
builder.Configuration.Sources.Clear();
builder.Configuration
    .AddJsonFile("sharedsettings.json", optional: true, reloadOnChange: true)
    .AddJsonFile("appsettings.json", optional: true, reloadOnChange: true)
    .AddJsonFile($"appsettings.{builder.Environment.EnvironmentName}.json", optional: true, reloadOnChange: true)
    .AddEnvironmentVariables();

if (builder.Environment.IsDevelopment())
{
    builder.Configuration.AddUserSecrets<Program>();
}
```

2. Забезпечте належне розташування параметра ConnectionString та коректну обробку різних значень для середовищ Development та Production. (max - 10 балів)

secrets.json:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=(localdb)\\MSSQLLocalDB;Database=DevDB;Trusted_Connection=True;"
  },
```

3. Створіть строго типізоване налаштування всієї ієрархії параметрів конфігурації. Додайте у контейнер DI застосунку сервіс конфігурації з життєвим циклом Singleton. Інжектуйте сервіс конфігурації через конструктор у контролері та використайте його для виведення у Footer інтерфейсу параметрів із завдання 1. (max - 25 балів)

Створюємо клас MyConfiguration:

```
namespace WebApplicationData.Models.Configurations
{
    4 references
    public class MyConfiguration
    {
        1 reference
        public string? ApplicationName { get; set; }
        1 reference
        public string? ApiKey { get; set; }
        1 reference
        public ConnectionStrings? ConnectionStrings { get; set; }
        1 reference
        public CustomSettings? CustomSettings { get; set; }
    }

    1 reference
    public class ConnectionStrings
    {
        1 reference
        public string? DefaultConnection { get; set; }
    }

    1 reference
    public class CustomSettings
    {
        1 reference
        public string? Message { get; set; }
    }
}
```

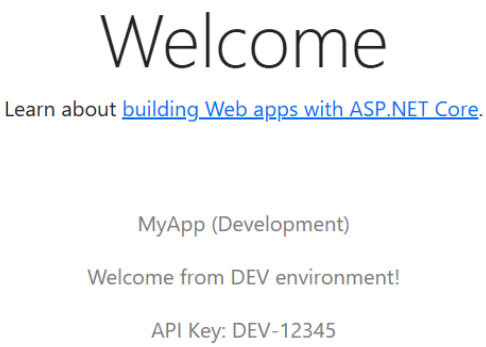
Program.cs:

```
var myConfig = builder.Configuration.Get<MyConfiguration>();
if (myConfig == null)
{
    throw new InvalidOperationException("Configuration object 'MyConfiguration' could not be loaded.");
}
builder.Services.AddSingleton(myConfig);
```

_Layout.cshtml:

```
<footer style="text-align:center; color:gray; margin-top:30px;">
    <p>@MyConfig.ApplicationName</p>
    <p>@MyConfig.CustomSettings?.Message</p>
    <p>API Key: @MyConfig.ApiKey</p>
</footer>
```

Результат:



4. Забезпечте конфігурацію параметром ApiKey. Забезпечте використання різних значень для середовища розробки та промислового середовища. (max-5 балів)

secrets.json:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=DevDB;Trusted_Connection=True;"
  },
  "ApiKey":  "DEV-12345"
}
```

```
$env:ASPNETCORE_ENVIRONMENT="Production"
$env:ApiKey="PROD-ABCDE"
$env:ConnectionStrings__DefaultConnection="Server=prodserver;Database=ProdDB;User Id=sa;Password=StrongPass123;"
```

5. Ознайомтеся з теоретичними основами middleware: що таке конвеєр обробки запитів, як працює делегування next, які є типи middleware. Наведіть приклади системного та користувацького middleware. (max - 10 балів)

Middleware — це програмний компонент, який збирається в конвеєр обробки запитів (request pipeline) для обробки HTTP-запитів та відповідей.

Конвеєр обробки запитів (Request Pipeline) — це послідовність Middleware-компонентів, розташованих один за одним. Кожен HTTP-запит, що надходить до застосунку, проходить через цей конвеєр, а відповідь повертається у зворотному порядку.

- **Налаштування конвеєра** відбувається у файлі `Program.cs` за допомогою методів розширення `app.Use...` (наприклад, `app.UseRouting()`, `app.UseAuthentication()`).

Як працює делегування `next`?

Кожен компонент Middleware у конвеєрі отримує посилання на наступний компонент у послідовності. Це посилання називається делегатом `next`.

- Функція `next`: Коли Middleware завершує свою роботу (наприклад, виконав логування або аутентифікацію), він викликає `await next(context);`, щоб передати управління та HTTP-контекст наступному компоненту в конвеєрі.
- Коротке замикання (Short-Circuiting): Якщо компонент Middleware вирішує, що запит не повинен продовжувати обробку (наприклад, виявлено помилку, або запит обслужено статичним файлом), він не викликає `next`. Замість цього він самостійно формує відповідь і повертає її, змушуючи конвеєр "замикатися" і відповідь починає рухатися у зворотному напрямку.

Middleware можна класифікувати за їхньою роллю:

1. Системне (Built-in) Middleware: Компоненти, які поставляються разом із фреймворком ASP.NET Core і виконують базові та критичні функції застосунку.
2. Користувацьке (Custom) Middleware: Компоненти, створені розробником для виконання специфічної логіки, що потрібна лише цьому застосунку.

Тип Middleware	Призначення	Приклад у коді (Program.cs)

Системне	**Обробка винятків** та помилок. У Dev-середовищі відображає детальні помилки.	`app.UseDeveloperExceptionPage();`
Системне	**Маршрутизація**; визначає, який кінцевий об'єкт (Controller/Razor Page) відповідає запиту.	`app.UseRouting();`
Системне	**Аутентифікація** та ідентифікація користувача.	`app.UseAuthentication();`
Користувацьке	**Логування запитів**; вимірювання часу виконання запиту.	Клас `RequestTimeMiddleware` (потрібно створити)

Користувацьке	**Перевірка заголовків**; наприклад, додавання спеціального заголовка безпеки.	Клас SecurityHeadersMiddleware

6. Додати Partitioned Rate Limiting middleware, що надає різні привілеї (кількість запитів за хвилину) для автентифікованих та неавтентифікованих користувачів. В разі обмеження повертати статус 429 – Too Many Requests. (max - 20 балів)

Program.cs:

```
builder.Services.AddRateLimiter(options =>
{
    options.OnRejected = async (context, _) =>
    {
        context.HttpContext.Response.StatusCode = StatusCodes.Status429TooManyRequests;
        await context.HttpContext.Response.WriteAsync("Too many requests. Please try again later.");
    };

options.GlobalLimiter = PartitionedRateLimiter.Create<HttpContext, string>(httpContext =>
{
    if (httpContext.User.Identity?.IsAuthenticated == true)
    {
        var userId = httpContext.User.Identity?.Name ?? "unknown";
        return RateLimitPartition.GetFixedWindowLimiter(
            partitionKey: $"user:{userId}",
            factory: _ => new FixedWindowRateLimiterOptions
            {
                PermitLimit = 100,
                Window = TimeSpan.FromMinutes(1),
                QueueLimit = 0,
                QueueProcessingOrder = QueueProcessingOrder.OldestFirst
            });
    }
    else
    {
        var ip = httpContext.Connection.RemoteIpAddress?.ToString() ?? "unknown";
        return RateLimitPartition.GetFixedWindowLimiter(
            partitionKey: $"ip:{ip}",
            factory: _ => new FixedWindowRateLimiterOptions
            {
                PermitLimit = 5,
                Window = TimeSpan.FromMinutes(1),
                QueueLimit = 0,
                QueueProcessingOrder = QueueProcessingOrder.OldestFirst
            });
    }
});
});
```

7. Зафіксувати зміни у проєкті на GitHub. (max - 10 балів)
8. Оформити звіт. (max - 5 балів)