МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра мережевих та інтернет технологій

СУЧАСНІ ІНТЕРНЕТ ТЕХНОЛОГІЇ

ГЛОБАЛІЗАЦІЯ ТА ЛОКАЛІЗАЦІЯ ЗАСТОСУНКУ ASP.NET CORE

Лабораторне заняття № 5

Черкун Єви Сергіївни

Хід виконання роботи:

Завдання 1. Ознайомитися з теоретичними основами глобалізації та локалізації: поняття Culture, UICulture, RFC 4646, ISO 639, ISO 3166 (сайт Microsoft).

Глобалізація та локалізація у програмуванні

- **Глобалізація (Globalization, G11n)** – створення програми, яку можна легко адаптувати для різних мов і регіонів без змін у коді.
- **Локалізація (Localization, L10n)** – адаптація програми під конкретну мову та регіон (переклад текстів, формат дат і чисел, валюти).

Culture та UICulture (.NET)

- **Culture** – задає культурні параметри користувача (формат дат, чисел, валюти).
- **UICulture** – задає мову інтерфейсу (які ресурси та тексти відображаються).

Стандарти мови та країни

- **RFC 4646** – стандарт тегів мови, наприклад:
  - en-US → англійська (США)
  - uk-UA → українська (Україна)
- **ISO 639** – коди мов (en – англійська, uk – українська).
- **ISO 3166** – коди країн (US – США, UA – Україна).

Взаємозв’язок

- Теги мови об’єднують коди ISO 639 і ISO 3166:
- en-US → мова: en, країна: US
- uk-UA → мова: uk, країна: UA
- У .NET використовуються для налаштування Culture та UICulture, що дозволяє відображати інтерфейс під конкретного користувача.

Завдання 2. Додати до ASP.NET Core застосунку підтримку локалізації через `AddLocalization()` з вказанням папки ресурсів.

```
125
126     options.OnRejected = async (context, cancellatio
127     {
128         context.HttpContext.Response.StatusCode = 42
129         context.HttpContext.Response.ContentType = "
130
131         var executor = context.HttpContext.RequestSe
132         var viewResult = new Microsoft.AspNetCore.Mv
133         {
134             ViewName = "/Views/Shared/TooManyRequest
135         };
136
137         await executor.ExecuteAsync(
138             new Microsoft.AspNetCore.Mvc.ActionConte
139                 context.HttpContext,
140                 context.HttpContext.GetRouteData() ?
141                 new Microsoft.AspNetCore.Mvc.Abstrac
142             ),
143             viewResult
144         );
145     };
146 });
147
148 builder.Services.AddLocalization(options =>
149 {
150     options.ResourcesPath = "Resources";
151 });
```

Рисунок 5.1 – Включення локалізації з папкою Resources до проекту

Завдання 3. Створити ресурсні файли `.resx` для базової культури та щонайменше двох додаткових.

Створюємо файли для трьох мов – чеської, англійської та української, які надалі треба для кожної сторінки заповнити відповідні переклади.

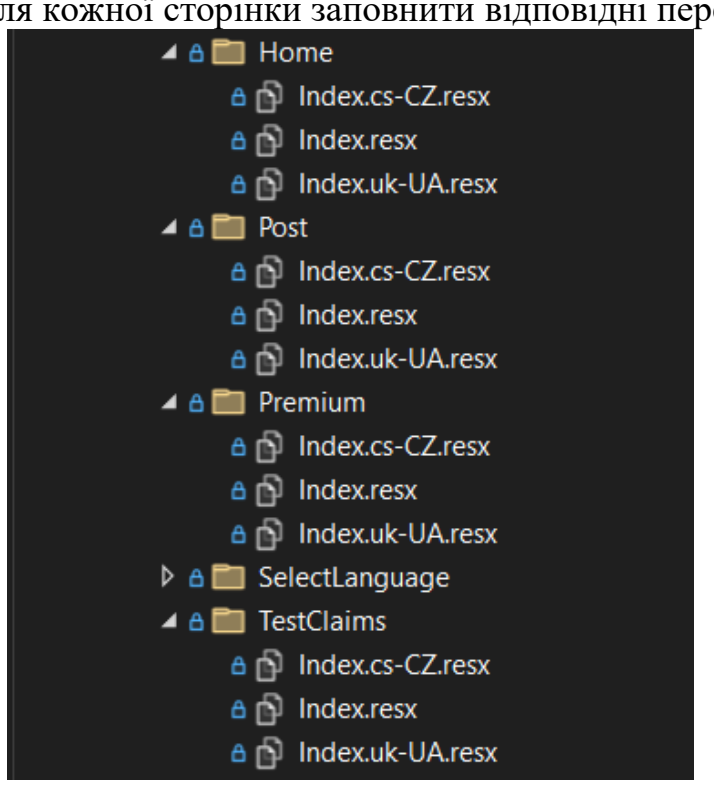

Рисунок 5.2 – Створені ресурсні файли

Name	Neutral Value	Neutral Comment	uk-UA
Available	This page is only available for user...		Ця сторінка доступна тільки кор...
Content	This is explicit content is only for p... \n You have access only be...		Це ексклюзивний контент для Pr... \n Ви маєте доступ до цієї...
ContentTitle	Premium content		Преміум контент
Title	Premium page		Преміум сторінка
Welcome	Welcome to the premium page!		Вітаємо на преміум сторінці!
YourHours	Your working hours:		Ваші робочі години:

Рисунок 5.3 – Приклад заповнення

Завдання 4. Налаштувати `RequestLocalizationOptions` з підтримкою щонайменше трьох культур. Встановити культуру за замовчуванням. Для цього додамо наступний код до конвеєра

```
var supportedCultures = new[] { "en-US", "uk-UA", "cs-CZ" };

builder.Services.Configure<RequestLocalizationOptions>(options =>
{
    options.SetDefaultCulture("en-US")
    .AddSupportedCultures(supportedCultures)
    .AddSupportedUICultures(supportedCultures);
});
```

Рисунок 5.4 – Налаштування підтримки трьох культур

Завдання 5. Додати middleware `UseRequestLocalization()` у правильному місці конвеєра запиту. Пояснити, чому порядок має значення.

Перед обробкою ендпоїнтів, маршрутів та інших запитів додамо проміжне програмне збезпечення UseRequestLocalization . Чому порядок має значення? Воно розміщене на цьому етапі, щоб локалізація застосовувалась до запитів ще до формування відповідей.

```
var app = builder.Build();

app.UseRequestLocalization();

if (app.Environment.IsDevelopment())
{
    app.UseMigrationsEndPoint();
}
else
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRateLimiter();

app.UseRouting();
```

Рисунок 5.5 – Включення локалізації в конвеєр

Завдання 6. Реалізувати типізовану локалізацію у контролері через IStringLocalizer<T> та у Razor View через IViewLocalizer.

Для цього, замінимо в Razor відображеннях захардкоджені фрази за допомогою отримання із ресурсних файлів за ключем

```
@{
    ViewData["Title"] = Localizer["Title"];
}
@inject Microsoft.AspNetCore.Mvc.Localization.IViewLocalizer Localizer

<h1>@ViewData["Title"]</h1>

<div class="alert alert-success">
    <h2>@Localizer["Welcome"]</h2>
    <p>@Localizer["Available"]</p>
    @if (ViewData["WorkingHours"] != null)
    {
        <p><strong>@Localizer["YourHours"]</strong> @ViewData["WorkingHours"]</p>
    }
</div>

<div class="card">
    <div class="card-body">
        <h3 class="card-title">@Localizer["ContentTitle"]</h3>
        <p class="card-text">
            @Localizer["Content"]
        </p>
    </div>
</div>
```

Рисунок 5.6 – Використання локалізації в шаблоні відображення

Завдання 7. Реалізувати перемикання мов, що встановлює культуру через cookies. Перевірити збереження обраної культури між запитами та сесіями.

Завантаживши Chrome розширення та змінивши за допомогою нього значення заголовку Accept-Language на cs для localhost домену за портом, зможемо перевірити чеську локалізацію інтерфейсу. За належного заповнення ресурсних файлів, відкривши Premium сторінку, маємо:

Додавши Cookies провайдер в RequestLocalizationOptions, обрана мова буде зберігатись в кеші браузера. Реалізуємо ручне перемикання мов в інтерфейсі додатку

Реалізуємо контролер та view, які дозволять користувачеві вибирати мову, зберігаючи вибір у Cookie.

```
using Microsoft.AspNetCore.Localization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Options;
using System;
using System.Linq;
using System.Globalization;

namespace WebAppCore.Controllers
{
    public class SelectLanguageController : Controller
    {
        private readonly IOptions<RequestLocalizationOptions> LocOptions;

        public SelectLanguageController(IOptions<RequestLocalizationOptions> locoptions)
        {
            LocOptions = locoptions;
        }

        public IActionResult Index(string returnUrl)
        {
            ViewData["ReturnUrl"] = returnUrl;

            var cultureItems = LocOptions.Value.SupportedUICultures?.ToList();
            return View(cultureItems);
        }
```

```
        [HttpPost]
        public IActionResult SetLanguage(string cultureName, string returnUrl)
        {
            Response.Cookies.Append(
                CookieRequestCultureProvider.DefaultCookieName,
                CookieRequestCultureProvider.MakeCookieValue(new RequestCulture(cultureName)),
                new CookieOptions { Expires = DateTimeOffset.UtcNow.AddYears(1) }
            );

            if (string.IsNullOrEmpty(returnUrl))
            {
                return RedirectToAction("Index", "Home");
            }
            else
            {
                return LocalRedirect(returnUrl);
            }
        }
    }
}
```

Рисунок 5.7 - Створення контролера SelectLanguageController

Далі створюємо папку у Views/SelectLanguage. Це View відобразить кожну підтримувану культуру як кнопку-форму, що викликає дію SetLanguage контролера.

Щоб кнопка перемикання була доступна на всіх сторінках, додайте посилання на SelectLanguage/Index у ваш основний шаблон (_Layout.cshtml), наприклад,у навігаційну панель.

```
WebAppCore
@using System.Globalization
@using Microsoft.AspNetCore.Mvc.Localization
@model IEnumerable<CultureInfo>
@inject IViewLocalizer Localizer

@{
    ViewData["Title"] = Localizer["lblChangeLanguage"];
}

<h2>@ViewData["Title"]</h2>

<div class="row">
    @foreach (var culture in Model)
    {
            <div class="col-md-3">
                <form asp-action="SetLanguage" method="post"
                      asp-route-cultureName="@culture.Name"
                      asp-route-returnUrl="@ViewData["ReturnUrl"]">
                  <button class="btn btn-link" type="submit">
                  @culture.DisplayName
                  </button>
                </form>
            </div>
    }
</div>

@* Блок кнопки "Назад" *@
@if (ViewData["ReturnUrl"] != null)
{
        <div class="mt-3">
            <a href="@ViewData["ReturnUrl"]" class="btn btn-secondary">@Localizer["lblGoBack"]</a>
        </div>
}
```

Рисунок 5.8 - Створення View для перемикання мови (Index.cshtml)

Завдання 8. Реалізувати підтримку параметрів URL (`?culture=en-US`) для вибору культури. Перевірити зміну мови.

```
builder.Services.AddLocalization(options =>
{
    options.ResourcesPath = "Resources";
});

var supportedCultureNames = new[] { "en-US", "uk-UA", "cs-CZ" };

builder.Services.Configure<RequestLocalizationOptions>(options =>
{
    var cultureInfos = supportedCultureNames
        .Select(name => new CultureInfo(name))
        .ToList();

    options.DefaultRequestCulture = new RequestCulture("en-US");
    options.SupportedCultures = cultureInfos;
    options.SupportedUICultures = cultureInfos;

    options.RequestCultureProviders = new List<IRequestCultureProvider>
    {
        new QueryStringCultureProvider(supportedCultureNames),
        new CookieRequestCultureProvider(),
        new AcceptLanguageHeaderRequestCultureProvider()
    };
});
```

Рисунок 5.9– Фрагмент Program.cs з підключенням QueryStringCultureProvider і налаштуванням порядку провайдерів у RequestLocalizationOptions

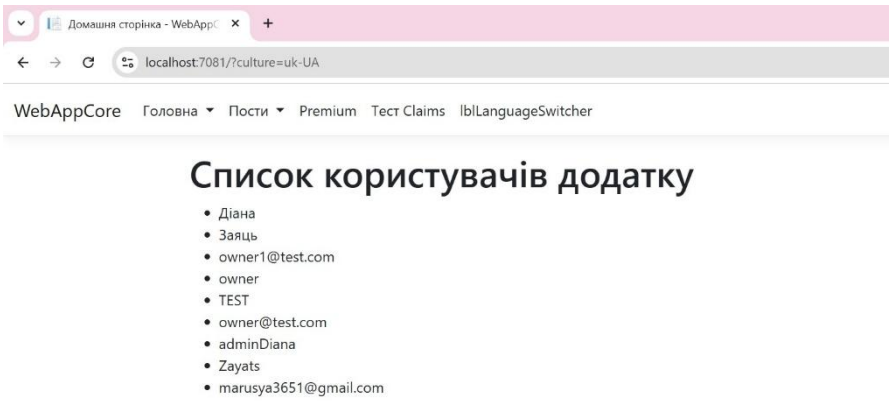

Рисунок 5.10 – Приклад роботи перемикання культури через URL: сторінка застосунку при виклику з параметром ?culture=uk-UA

Завдання 9. Реалізувати локалізацію форматів дат, чисел і валют у залежності від обраної культури.

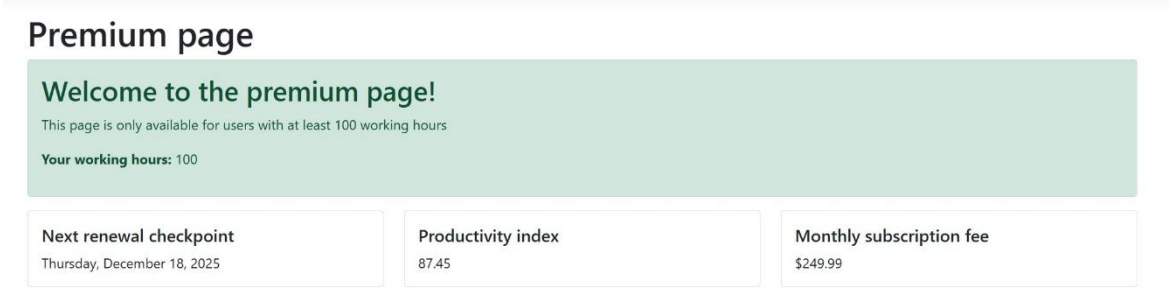

Рисунок 5.11 – Сторінка Premium при культурі en-US (формати дати, числа та

валюти для США)

Рисунок 5.12 – Та сама сторінка Premium при культурі cs-CZ (інші роздільники та позначення валюти Kč)

Висновок:

У процесі виконання лабораторної роботи було досліджено та практично реалізовано механізми авторизації в ASP.NET Core. Було створено політики доступу на основі клеймів користувача, що забезпечують гнучке керування доступом до сторінок і ресурсів веб-застосунку. Крім того, опрацьовано імперативну авторизацію з перевіркою ресурсів, яка дозволяє обмежувати дії користувача лише власними даними, наприклад, редагування власного поста на форумі. Набуті навички сприяють безпечній організації доступу користувачів до різних частин веб-застосунку та дають змогу реалізовувати складні правила контролю доступу.