Київський національний університет ім. Тараса Шевченка
Кафедра мережевих та інтернет технологій

Лабораторна робота № 5

Дисципліна: Сучасні Інтернет Технології
Тема: Глобалізація та локалізація застосунку ASP.NET CORE
Підготувала студентка МІТ-41
Пась Олександра

Хід виконання

2. Додати до ASP.NET Core застосунку підтримку локалізації через AddLocalization() з вказанням папки ресурсів.
Для цього завдання потрібно створити папку Resourses та додати підтримку локалізації в Program.cs.

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

рис. 5.1 - додавання AddLocalization

3. Створити ресурсні файли .resx для базової культури та щонайменше двох додаткових.
У папці, яку ми створили у попередньому завданні, створюємо папку Controllers і в ній створюємо3 файли .resx:

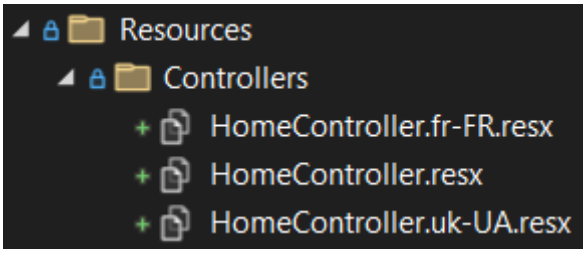


рис. 5.2 - створені файли

Це спеціальні файли, у яких зберігаються ключі та їхні переклади. Один і той самий ключ є у кожному файлі, але значення різне — відповідно до мови. Завдяки цьому застосунок може автоматично показувати користувачу текст тією мовою, яка зараз вибрана. Іншими словами, .resx — це місце, де лежать переклади інтерфейсу.

Name	Neutral Value	Neutral Comment	uk-UA	fr-FR
lblChangeLanguage	Change language		Змінити мову	Changer la langue
Test	Hello World!		Привіт Світ!	Bonjour le monde!

рис. 5.3 - вміст файлу HomeController.resx

4. Налаштувати RequestLocalizationOptions з підтримкою щонайменше трьох культур. Встановити культуру за замовчуванням.
У цьому завданні ми налаштували список культур (мов), які підтримує застосунок, і встановили мову за замовчуванням.
RequestLocalizationOptions повідомляє системі, які культури існують, які з

них можна обирати та яку використовувати, якщо користувач не вибрав нічого. Без цього кроку застосунок не розумів би, між якими мовами можна перемикатися і яку культуру використовувати для форматування дат, чисел та текстів.

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

// Додати з підтримкою локалізації для View
builder.Services.AddControllersWithViews()
    .AddViewLocalization()  /
    .AddDataAnnotationsLocalization();
```

рис. 5.4 - RequestLocalizationOptions в program.cs

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

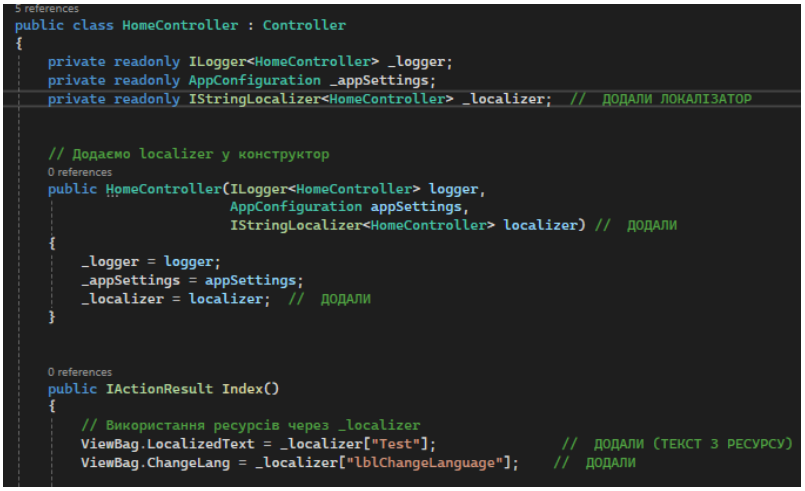
Тут ми додали middleware, який відповідає за визначення мови при кожному запиті. Це потрібно, щоб перед обробкою сторінки система вже знала, яку культуру використовувати. Порядок важливий: якщо додати middleware пізніше — сторінки вже будуть завантажені без перекладу. Тому ми ставимо його на початку конвеєра, щоб локалізація працювала на всіх сторінках одразу.

```
app.UseRequestLocalization();
```

рис. 5.5 - middleware UseRequestLocalization()

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

У цьому завданні ми додали можливість отримувати перекладені тексти у коді контролера та у Razor-виглядах. IStringLocalizer дозволяє контролеру отримувати переклад за ключем, а IViewLocalizer робить те саме у .cshtml файлах. Це означає, що замість жорстко прописаних текстів у коді ми тепер використовуємо ключі, а реальний текст береться з .resx-файлів відповідної мови. Завдяки цьому інтерфейс стає багатомовним без дублювання коду.



```
5 references
public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;
    private readonly AppConfiguration _appSettings;
    private readonly IStringLocalizer<HomeController> _localizer;  //  ДОДАЛИ ЛОКАЛІЗАТОР


    // Додаємо localizer у конструктор
    0 references
    public HomeController(ILogger<HomeController> logger,
                          AppConfiguration appSettings,
                          IStringLocalizer<HomeController> localizer) //  ДОДАЛИ
    {
        _logger = logger;
        _appSettings = appSettings;
        _localizer = localizer;  //  ДОДАЛИ
    }


    0 references
    public IActionResult Index()
    {
        // Використання ресурсів через _localizer
        ViewBag.LocalizedText = _localizer["Test"];              //  ДОДАЛИ (ТЕКСТ З РЕСУРСУ)
        ViewBag.ChangeLang = _localizer["lblChangeLanguage"];    //  ДОДАЛИ
```

рис. 5.6 - HomeController

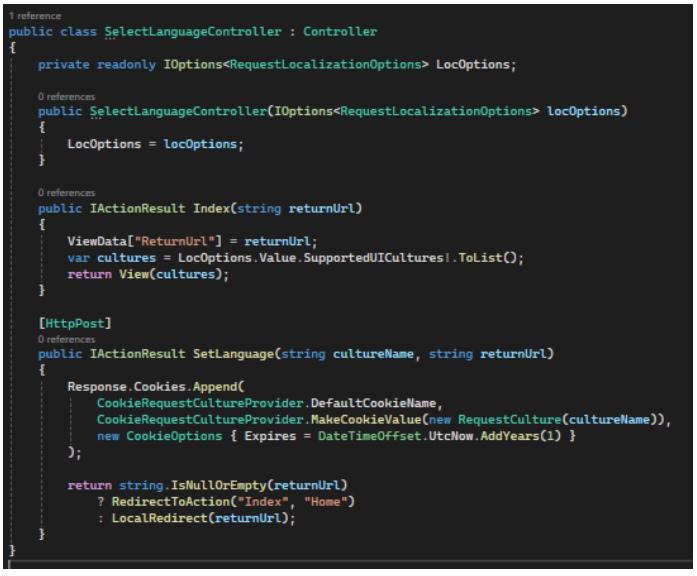
```
@inject Microsoft.AspNetCore.Mvc.Localization.IViewLocalizer Localizer
```

```
<h4>@ViewBag.LocalizedText</h4>
<p>@Localizer["lblChangeLanguage"]</p>
```

рис. 5.7 - index.cshtml

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

У цьому завданні ми реалізували механізм, який дозволяє користувачу змінювати мову вручну. Коли користувач обирає мову, ми записуємо її у cookie. Завдяки цьому вибір зберігається між перезавантаженнями сторінок і навіть після закриття браузера. Це робить інтерфейс зручнішим, тому що користувачеві не потрібно вибирати мову кожен раз заново.



```
1 reference
public class SelectLanguageController : Controller
{
    private readonly IOptions<RequestLocalizationOptions> LocOptions;

    0 references
    public SelectLanguageController(IOptions<RequestLocalizationOptions> locOptions)
    {
        LocOptions = locOptions;
    }

    0 references
    public IActionResult Index(string returnUrl)
    {
        ViewData["ReturnUrl"] = returnUrl;
        var cultures = LocOptions.Value.SupportedUICultures!.ToList();
        return View(cultures);
    }

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

        return string.IsNullOrEmpty(returnUrl)
            ? RedirectToAction("Index", "Home")
            : LocalRedirect(returnUrl);
    }
}
```

рис. 5.8 - SelectLanguageController

```
<h2>@Localizer["lblChangeLanguage"]</h2>

<div>
    @foreach (var culture in Model)
    {
        <form method="post"
              asp-action="SetLanguage"
              asp-route-cultureName="@culture.Name"
              asp-route-returnUrl="@ViewData["ReturnUrl"]">

            <button class="btn btn-link">@culture.DisplayName</button>
        </form>
    }
</div>
```

рис. 5.9 - index.cshtml

Перевіримо роботу запустивши проект та ввівши https://localhost:7131/SelectLanguage?returnUrl=/

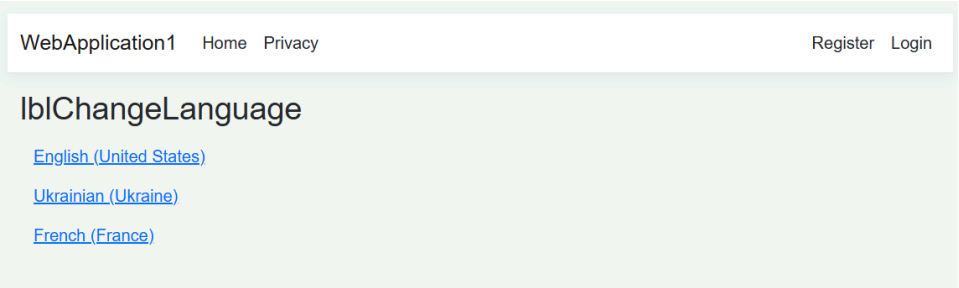


рис. 5.10 - вибір мови

Обираємо мову, яку хочемо наприклад українську.

Маємо сторінку українською:

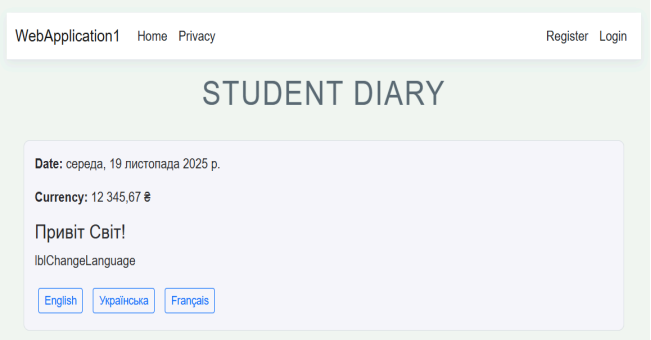


рис. 5.11 - приклад роботи

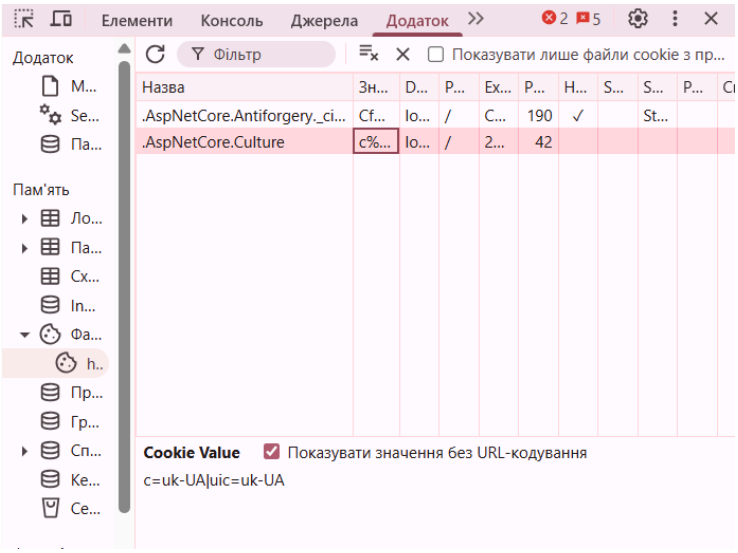


рис. 5.12 - cookie

Якщо зупинити проект і запустити знову, сторінка буде українською мовою.

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

Це завдання додає можливість перемикати мовну культуру прямо через адресу сторінки, наприклад ?culture=uk-UA. Такий спосіб корисний для швидкого тестування та для зовнішніх посилань, які можуть вказувати на

певну мовну версію сайту. Коли параметр передається у URL, застосунок автоматично перемикається на відповідну культуру.

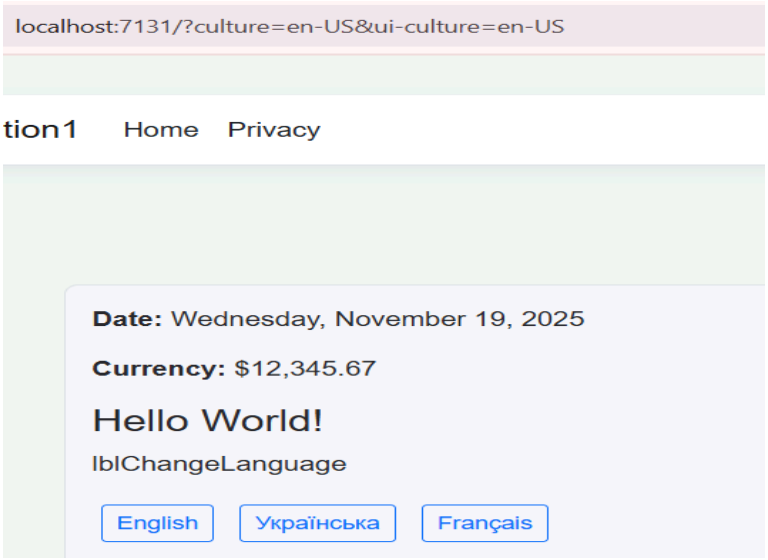


рис.5.13 - приклад роботи

У цьому завданні ми явно нічого не робили, тому що у завданні 4 ми автоматично підключили це завдяки RequestLocalizationOptions.

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

У цьому завданні ми перевірили, як зміна культури впливає на форматування дат і чисел. Різні культури по-різному показують валюту, роздільники чисел та формат дати. Наприклад, англійська культура використовує $ і крапку, а українська — ₴ і кому. Це демонструє, що локалізація впливає не тільки на переклад текстів, а й на стандартне відображення даних.

```
<p><strong>Date:</strong> @DateTime.Now.ToLongDateString()</p>
<p><strong>Currency:</strong> @(12345.67.ToString("C"))</p>
```

рис. 5.14 - index.cshtml

На рис. 5.11 та 5.13 можна побачити, що відповідно до обраної мови відбувається форматування.

Висновок: У цій лабораторній ми додали в застосунок підтримку кількох мов, створили файли перекладів, налаштували вибір культури та зробили можливість перемикати мову через cookies і URL. Переклади почали працювати і в контролерах, і у виглядах, а формат дат і чисел змінюється відповідно до вибраної культури. Застосунок став повністю багатомовним і зручним для користувачів різних країн.