Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.738.5
DOI: https://doi.org/10.24867/01BE23Oljaca

INTERNACIONALIZACIJA I LOKALIZACIJA WEB APLIKACIJE RAZVIJENE KORIŠĆENJEM ASP.NET CORE MVC FRAMEWORK

INERNATIONALIZATION AND LOCALIZATION OF WEB APPLICATION DEVELOPED USING THE ASP.NET CORE MVC FRAMEWROK

Marko Oljača, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U okviru rada je izvršena analiza lokalizacije i internacionalizacije u okviru Microsoft tehnologija. Kreirana je web aplikacija korišćenjem ASP.NET Core MVC frameworka, a u procesu pravljenja iste prezentovani su pojmovi lokalizacije i internacionalizacije. Pošto je .NET Core sam po sebi relativno novi koncept u okviru Microsoft tehnologija početni deo rada se bavi samim razvojem te tehnologije. Zatim je objašnjeno kako u okviru .NET Core korišćenjem MVC frameworka možemo da kreiramo Web aplikaciju. Na kraju se u radu objašnjavaju tehnologije i alati koji se koriste za internacionalizaciju i lokalizaciju.*

Ključne reči: *.NET Core, MVC .NET Core, Internacionalizacija .NET Core web aplikacije, Globalizacija .NET Core aplikacije*

Abstract – *In this thesis theme was deep dive into localization and internationalization analysis within Microsoft technologies. Web application was created using ASP.NET Core MVC framework, and though the process of making it concepts of localization and internationalization were presented. Since .NET Core itself is a relatively new concept within Microsoft technology in the early part of the work is explained development of this technology. Then explained how to create Web application using the MVC framework in .NET Core. Finally part is dedicated to explanation of technologies and tools used for localization and internationalization.*

Keywords: *.NET Core, MVC .NET core, Internatiolization .NET Core Web application, Globalization .NET Core Web application*

1. UVOD

U savremenom svetu informacionih tehnologija omogućeno je da se lako povezujemo sa ljudima iz različitih kultura, koji koriste različite jezike i generalno komuniciraju na različite načine.

S tim na umu zadatak koji se nameće prilikom razvoja aplikacije jeste da ih pravimo tako da mogu sa lakoćom da menjaju svoj izgled na taj način da je mogu koristiti korisnici iz različitih kultura.

Zadatak rada jeste dublja analiza koncepata lokalizacije i internacionalizacije u okviru *Microsoft* tehnologija. Cilj je da se napravi web aplikacija korišćenjem ASP.NET Core MVC *frameworka* a u procesu pravljenja iste da se prezentuju pojmovi lokalizacije i internacionalizacije.

2. POJAM KULTURE U INFORMACIONIM TEHNOLOGIJAMA

Kultura je skup pravila i podataka koja su specifična za kombinaciju jezika i geografskog područja koje kultura jedinstveno reprezentuje.

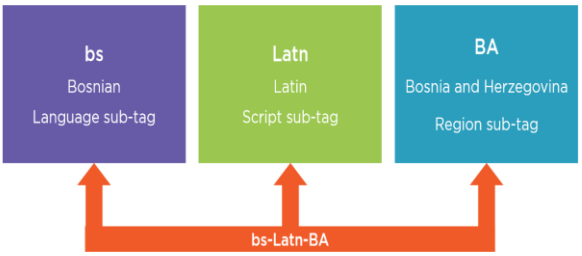
Locale = Culture u .NET okruženju local je sinonim za kulturu.

.Net Core ima podršku za različite kulture, tako da programeri ne moraju da se razumeju u pravila za specifične kulture.

Svaka kultura ima specifičan ***IETF Language Tag*** koji je jedinstveno obeležava. Primer jedno taga: **en-US**, kasnije će tag biti detaljno objašnjen.

IETF – Internet Egnineering Task Force su zaduženi za kreiranje i održavanje raznih internet standarda. Takođe su zaduženi za pojam internacionalizacije na internetu.

IETF Language Tag je identifikator jedne kulture koji je specificiran od strane **IETF.**



Slika 1. Struktura IETF *language tag-a*

2.1.POJAM KUTLURE u .NET

Skoro sve vezano za internacionalizaciju i18n u .NET Core se vrti oko klase **CultureInfo**. Ova klasa je postajala u .NET-u i portovana je na .NET Core i definisana je u System.Globalization modulu. Sve što nam je potrebno za

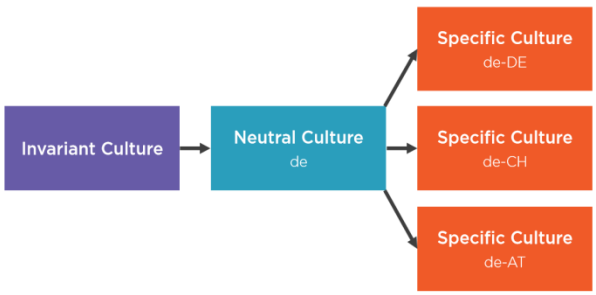
---

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Branko Milosavljević, red. prof.

korišćenje jedne kulture možemo da uradimo kreiranjem jedne instance pomenute klase.

CultureInfo ci = new CultureInfo(“en-US”);

Ono što je specifično za .NET Core je uvođenje dva pojma koji opisuju kulturu: **Culture** i **UI culture.** Oba pojma su intance CultureInfo klase, ali su odgovorni za različite aspekte internacionalizacije.
UI Culture je zadužena za
Prikazane tekstove
Culture je zadužena za
Format vremena i datuma
Format brojeva
Formatiranje vrednosti iskazanih u valutama
Redosled sortiranja
Velika i mala slova
Poređenje stringova
Ne postoji pravilo da **Culture** i **UI Culture** moraju imati istu vrednost, čak je ideja uvođenja dva različita pojma da bi se omogućila veća sloboda u razvoju. Odluka koje će **Culture** i **UI Culture** biti podržane i u kakvoj kombinaciji je procena koja se donosi u skladu sa biznis logikom.
Struktura samog taga ukazuje i na hijerarhiju kulture. Da bismo objasnili pojam hijerarhije kulture, moramo da uvedemo još par pojmova.
Neutralna kultura je kultura koja ima specificiran jezik, a nema specificiranu regiju ili pismo. Npr za Nemačku neutralna kultura je tag **de.** U samom C# postoji flag koji ukazuje na to da li je kultura neutralna. .IsNeutralCulture nam daje informaciju o tome da li je trenutna kultura neutralna kultura. Neutralna kultura nam omogućuje način da ipak podržimo veći broj kultura za specifičan jezik.
Invariant Cutlure je kultura bez taga, asocirana je za engleski jezik. Intanca invarijantne kulture se pravi na sledeći CutlureInfo.InvariantCulture. Ona nam može koristiti kada želimo da baratamo podacima koji nisu u relaciji sa bilo kojom specifičnom kulturom. Svaka kultura e hijerarhijski uvezana.



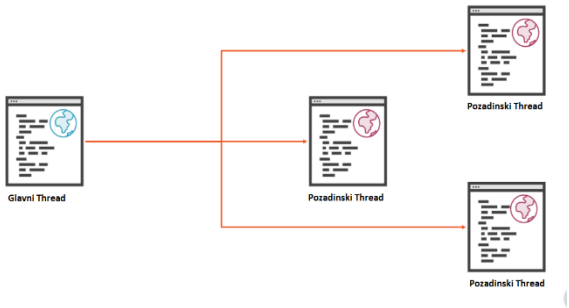
Slika 2. Hijerarhija kultura

2.2 KULTURE I MULTITHREADING

Svaka nit ima informaciju o kulturi pod kojom je ona pokrenuta, podaci o kulturi se mogu dobiti na sledeći način:

CultureInfo cultureInfo =
Thread.CurrentThread.CurrentCulture;

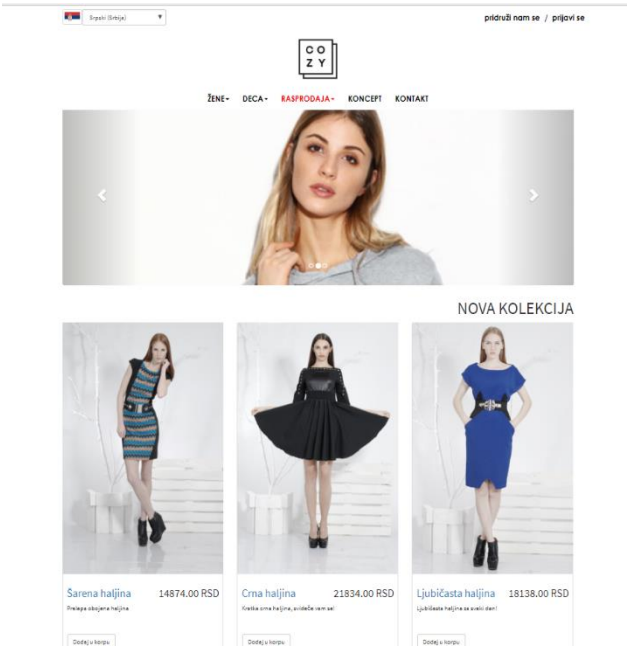
U slučaju kada se iz glavne niti kreira pozadinska nit, pozadinska nit nasleđuje kulturu od glavne niti. Svaka dalja promena kulture glavnog niti ili pozadinske niti neće uticati na kulturu niti kojoj nije promenjena kultura. Odnosno obe niti su posle kreiranja autonomne.
Nasleđivanje kulture prilikom kreiranja niti možemo zaobići postavljanjem default kulture za nit pre nego što je kreirana iz glavne niti. Property koji to omogućuje je DefaulThreadCurrentCutlure.



Slika 3. Kulture u nitima

3. SPECIFIKACIJA SISTEMA

Sistem koji se razvija u okviru ovog rada je web aplikacija za *on line* modnu prodavnicu. Nazvaćemo je *Master Fashion Store*. Ono što je pogodno za primer lokalizacije je to što u okviru rada sa prodavnicom je da moramo da obraćamo pažnju na rad sa različitim jezicima, apoenima i datumima. Na taj način se može slikovito prokazati svi aspekti tehnologije koje su potrebni za ovaj rad.



Slika 3. Izgled početne strane

4. IMPLEMENTACIJA SISTEMA

U ovom poglavlju ćemo se baviti implementacijom lokalizacije u referentnom projektu čime ćemo pokazati na koji način se implementira lokalizacija na projektima koji su implementirani korišćenjem MVC .Net Core-a *framework-a.*
Lokalizacija je u MVCu-u, kao i većina stvari implementirana upotrebom *middlewera*. Zato moramo da uvedemo pojam *middlewera.*

Šta je middleware?

Middleware je software koji se dodaje u tok aplikacije tako da je odgovoran za rad sa *requests* i *response*. Svaka komponenta ima dve opcije:

- Bira da li *request* da prosledi sledećoj komponenti u redu
- Može da uradi neki posao pre nego što se pozove sledeća komponenta u toku

U referentnom projektu smo koristili *Request Localization Middleware*. Izdvojićemo najvažnije karakteristike ove implementacije:

- Vezana je za životni vek jednog *HTTP requesta*. Kao što naziv middelwarea ukazuje *request* lokalizaijca je process koji se dešava u toku životnog veka jednog *HTTP requesta*.
- Preferencije klijenta diktiraju koja se kultura koristi
- Separacija *thredova*. Svaki HTTP *request* se izvršava u različitom *thredu,* tako da je moguće da paralelno dva korisnika gledaju sadržaj, a da je taj sadržaj prikazan u različitim kulturama.
- Nova instance lokalitazatora se kreira po *requestu.*

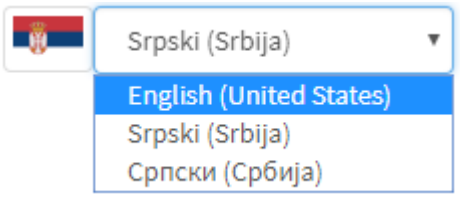
Prvo nam je potrebno da uključimo potrebne NuGet pakete, a zatim da uključimo sam *middelwhare* u aplikaciju. Od koda nam je potrebno samo ovo:

```
dependencies":
{
  ...,
  "Microsoft.AspNetCore.Localization":"1.1.1",
  ...
}
app.UseRequestLocalization(...);
```

Pošto smo dodali potrebne pakete, dodali smo i *middelwhare* koji nam je potreban za lokalizaciju. Sada nam ostaje da konfigurišemo njegovo ponašanje.

```
services.Configure<RequestLocalizationOptions>
(options =>
      {
  options.SupportedUICultures = new
List<CultureInfo>()
      {
          new CultureInfo("en-US"),
           new CultureInfo("sr-Latn-RS"),
           new CultureInfo("sr-Cyrl-RS")
      };
options.DefaultRequestCulture =
new RequestCulture("en-US");
});
```

Preko *Configure* metode i prosleđenog delegate možemo da konfigurišemo naš servis. Ovom prilikom smo definisali podržane kulture, kao i *default* kulturu za našu web aplikaciju.
Sa ovim koracima smo obezbedili da naša aplikacija podržava različite kulture i naveli smo koje kulture. Sada ćemo da vidimo na koji tačno način ćemo čuvati vrednost kulture i kako ćemo obezbediti da korisnik menja kulturu.



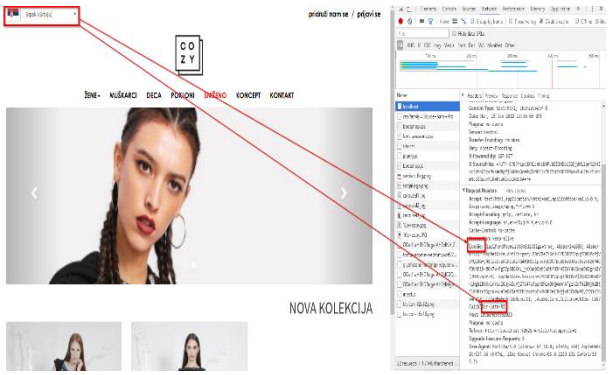
Slika 4. *Partial view* zadužen za izbor kutlure

Na slici 4. se vidi komponenta koja je zadužena za promenu kulture. Sa klijentske strane je to *Combobox* koji prikazuje podržane kulture. Ono što je bitno je da bi korisnicima bila jasnije koje kulture biraju imena država su napisana jezikom kulture koje biraju. Takođe sama zastava je lokalizovana. Sada ćemo da vidimo kontorler koji ima samo jednu metodu koja smešta aktivnu kulturu u *cookie* i tako nam obezbeđuje da u toku trajanja sesije korisnik koristi izabranu kulturu.

```
[HttpPost]
public IActionResult Set(string uiCulture, string
returnUrl, string returnParam)
                    {
          IRequestCultureFeature feature
HttpContext.Features.Get<IRequestCultureFeature>();

RequestCulture requestCulture =
 new RequestCulture(feature.RequestCulture.Culture,
   new CultureInfo(uiCulture));



string cookieValue =
          CookieRequestCultureProvider.MakeCookieVal
ue(requestCulture);
string cookieName =
          CookieRequestCultureProvider.DefaultCookieN
ame;
```

```
Response.Cookies.Append(cookieName, cookieValue);
    if (!String.IsNullOrEmpty(returnParam))
        {
            returnUrl = returnUrl + returnParam;
        }

        Return LocalRedirect(returnUrl);
}
```

Ovaj kod nam pokazuje kako se uzima izabrana kontrola od strane korisnika i smeštanje tog izbora u *cookie*.



Slika 5. Izbor kulture se smešta u cookie

4. ZAKLJUČAK

Master Fashion Store je studija slučaja na osnovu koje je prikazan proces lokalizacije i globalizacije. Osim same implementacije proces dizajna aplikacije koja u startu ima globalni pristup je ono što je bitno istaći. Internacionalizacija je jako ozbiljan proces kome se ponekad ne daje dovoljno pažnje u ranim fazama razvoja aplikacije, što dovodi do niza problema u kasnijim fazama. Ako smo svesni da je naša aplikacija internacionalnog karaktera, internacionalizacija mora biti uključena od samog starta razvoja.

Pošto je pristup arhitekturi razrađen uvidom u implementaciju je prikazano na koji način je u sklopu MVC .Net Core omogućeno realizovati internacionalizaciju. Videli smo alate i načine implementacije internacionalizacije. Bitan je i način projektovanja baze podataka koji u startu ima u vidu koji pojmovi se trebaju lokalizovati i kako.

Na osnovu iskustva u toku implementacije izdvojio bih par preporuka.

Preporuke za UI

- Zastava nije kultura

Jako je bitno da ne brkamo zastavu i kulturu, preporuka je da se naziv zemlje piše korišćenjem upravo pisma koje je specificiranu za tu kulturu. Nikako ne smemo pisati na engleskom nazive država.

- Autodetekcija jezika korisnika

Da bismo olakšali korisniku treba da vršimo predikciju jezika koji on koristi. U referentnoj aplikaciji se čita iz cookija, a ako nije podešeno setuje se u cookie-u.

- Jasan način da je korisnik svestan koji deo aplikacije je za promenu jezika

U referentnoj aplikaciji sve vreme je korisniku dostupna promena jezika, a zastava se koristi upravo da se korisniku pojasni čemu služi taj deo funkcionalnosti.

Prezentovan je način razmišljanja i alati koji su potrebni da se napravi aplikacija koja je spremna za svetsko tržište.

5. LITERATURA

[1] Microsoft dokumentacija, **https://docs.microsoft.com/en-us/globalization/software-internationalization**

[2] Gill Cleeren, Buidling your first ASP.NET Core Web Application, Pluralsight**,** **https://app.pluralsight.com/library/courses/aspdotnetcore-web-application-building/table-of-contents**

[3] Roland Gujit, Understanding ASP.NET Core Security, Pluralsight, **https://app.pluralsight.com/library/courses/asp-dot-net-core-security-understanding/table-of-contents**

[4] Admir Tuzović, ASP.NET Core Internationalization Deep Dive, Pluralsight**,**

https://app.pluralsight.com/library/courses/aspdotnetcore-internationalization-deep-dive/table-of-contents

Kratka biografija:

Marko Oljača rođen je 29.07.1983. godine u Novom Sadu. Osnovnu školu „Prva Vojvođanska Brigada“ završio je 1998. godine. Gimnaziju „Jovan Jovanović Zmaj“ u Novom Sadu završio je 2002. godine. Iste godine upisao se na Fakultet tehničkih nauka, odsek Računarstvo i automatika. Školske 2002/2003. godine upisao se na smer Računarske nauke i informatika. Osnovne studije završio 2012 godine godine. Zaposlen u *Schneider Eletric DMS NS* 2012 – 2017. Od 2017 do danas zaposlen u *Enjoying, Belgrade*, Serbia. Položio je sve ispite predviđene planom i programom Master studija na smeru Računarske nauke i informatika.