Osnove web programiranja

Internacionalizacija sadržaja

Termin 6

Sadržaj

1. Potreba za internacionalizacijom sadržaja
2. Definisanje poruka
3. izvora poruka
4. Podešavanje konfiguracije
5. Korišćenje različitih poruka za različite korisnike

Potreba za internacionalizacijom sadržaja

Uvod

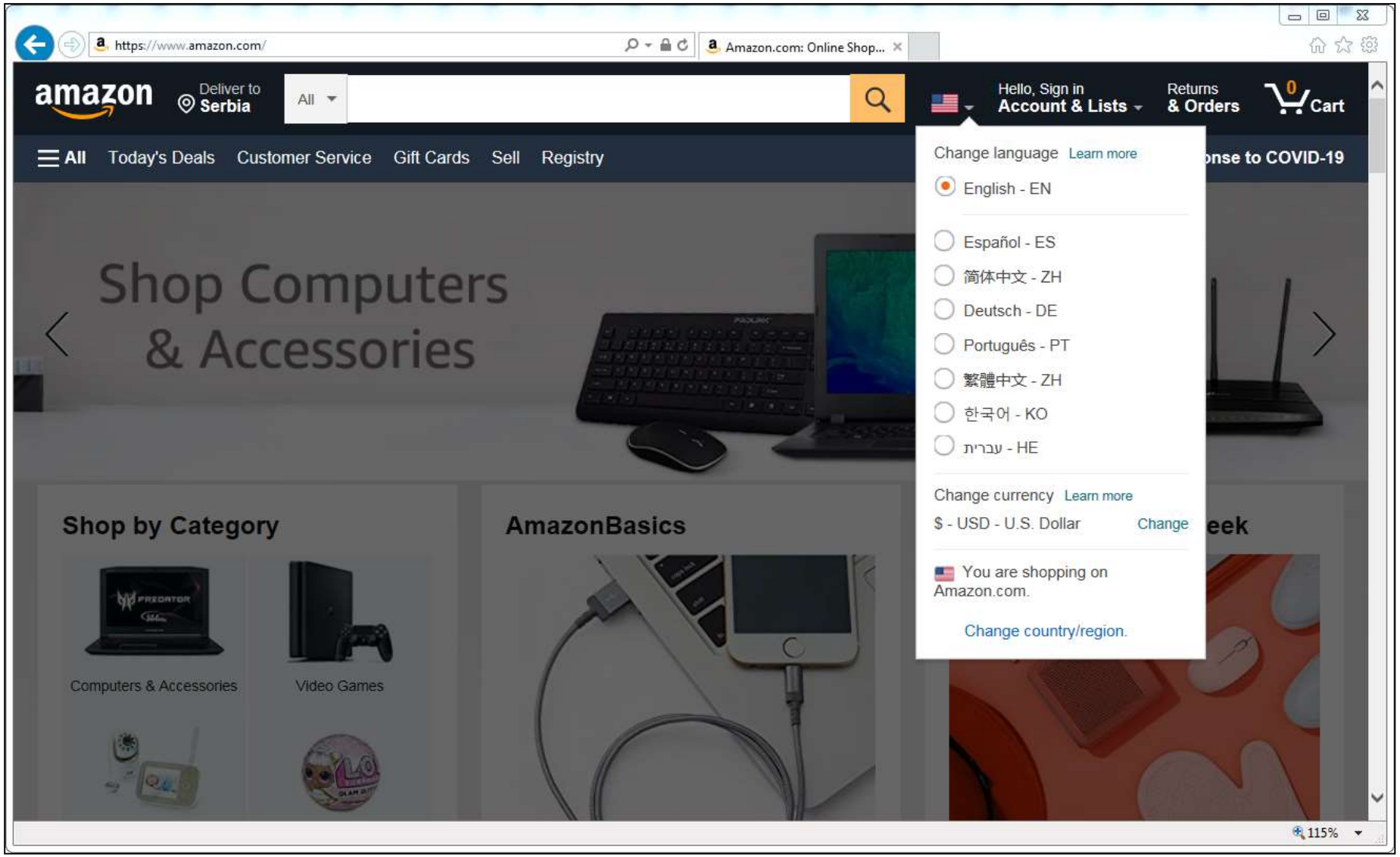
Internacionalizacijom sadržaja koji se dostavlja korisnicima, aplikacije se prilagođava potrebama korisnika koji dolaze iz različitih zemalja/regiona.

Na ovaj način aplikacija postaje dostupna široj populaciji korisnika.

- Ne znaju svi korisnici predefinisani jezik na kome je pravljena apliakcije

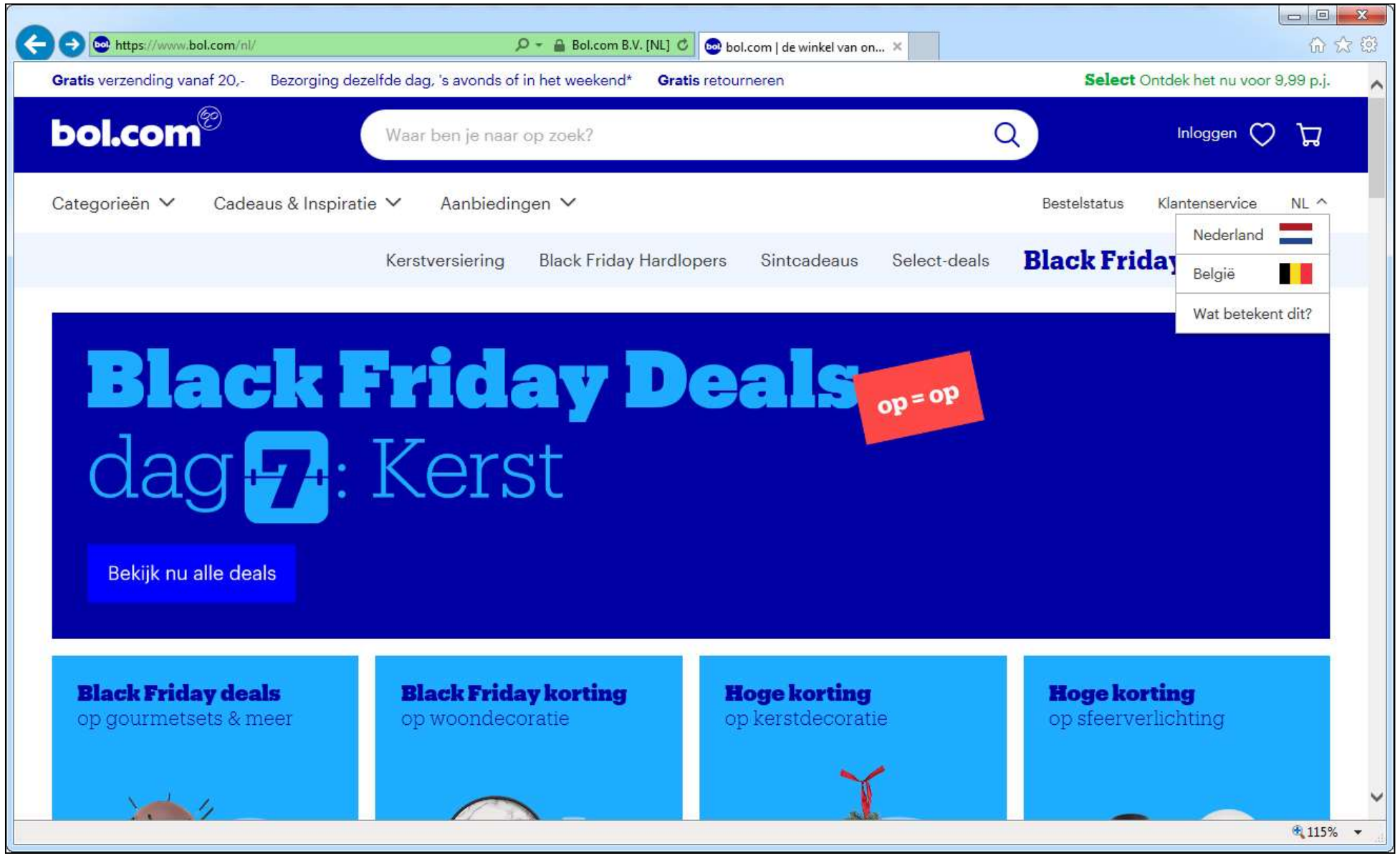
Potreba za internacionalizacijom sadržaja

Uvod



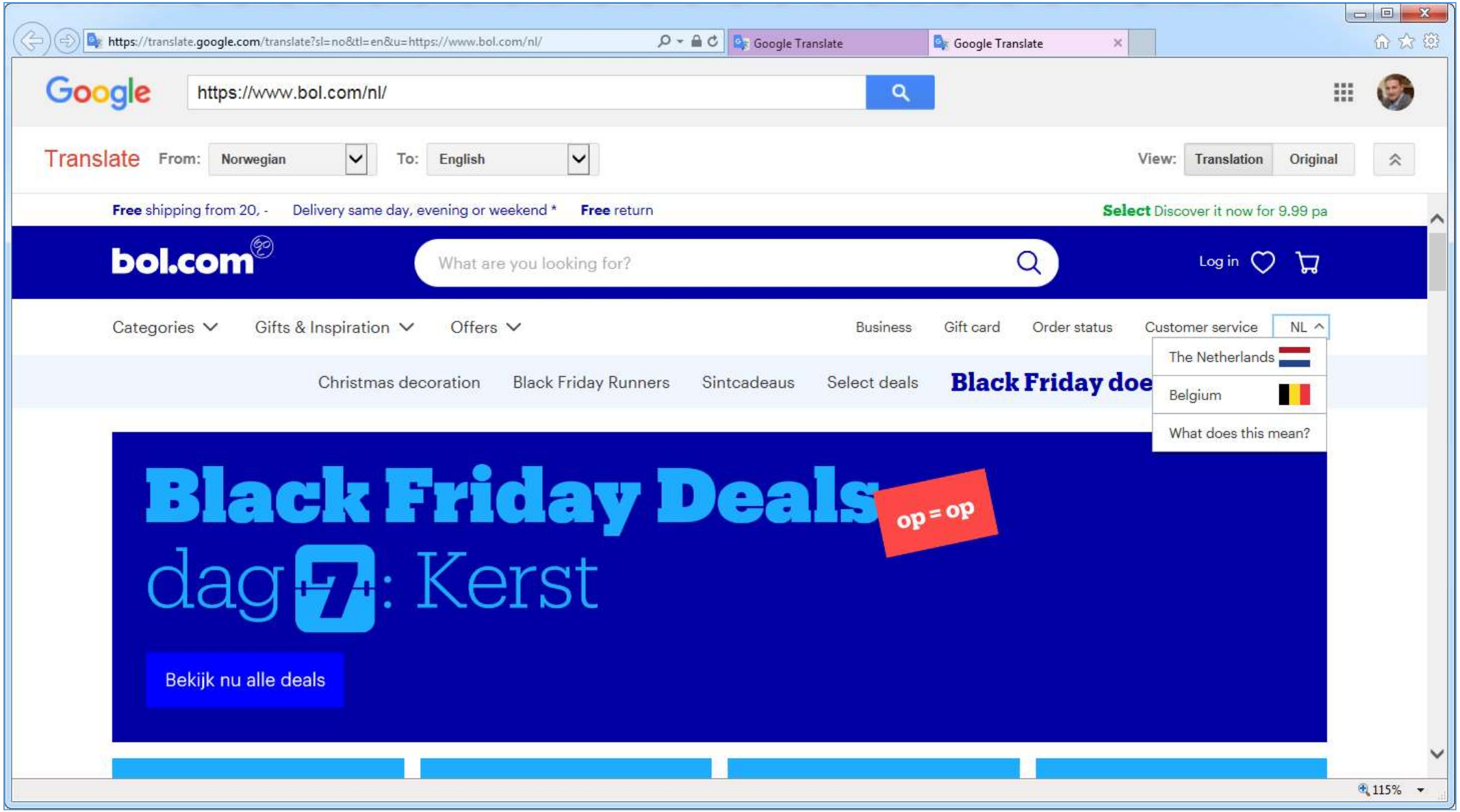
Potreba za internacionalizacijom sadržaja

Uvod



Potreba za internacionalizacijom sadržaja

Uvod



Potreba za internacionalizacijom sadržaja

Uvod

Osnovna ideja internacionalizacije sadržaja može se videti iz činjenice da različiti korisnici imaju potrebu da koriste aplikaciju na različitim jezicima/pismima.

Definisanje poruka

Datoteke u kojima se navode poruke

Sav tekstulani sadržaj koji se koristi za potrebe internacionalizacije piše se u property fajlovima, a oni se skladište u okviru **src/main/resources** direktorijuma.

Obično se kreira defaultni fajl pod nazivom **messages.properties** u kome se navodi predefinisani/osnovni set poruka koji se koristi ako aplikacija ne podržava poruku za odabrani jezik internacionalizacije.

Za svaki jezik internacionalizacije koji treba da podržava aplikacija se navodi set poruka u okviru zasebnog property fajla, pri čemu taj fajl sadrži **sufiks** koji predstvlja kod zemlje (npr. **messages_sr.properties**, **messages_de.properties**, **messages_nl.properties,**...)

Definisanje poruka

Izgled messages.properties datoteke

Sadržaj datoteke navodi se po principu **ključ=vrednost**

Ključ može da se sastojati iz više reči koje se odvajaju simbolom "." sa ciljem da se definiše hijerahija poruka u okviru samog fajla, a obično odgovara lokaciji labele u HTML fajlu za koju se definiše poruka

messages.properties

```
filmovi.index.title=Movies
filmovi.index.table.caption=Movies
filmovi.index.table.redBr=no.
filmovi.index.table.naziv=title
projekcije.index.table.caption=Projections
```

messages_sr.properties

```
filmovi.index.title=Filmovi
filmovi.index.table.caption=Filmovi
filmovi.index.table.brojStavke=r. br.
filmovi.index.table.naziv=naziv
projekcije.index.table.caption=Projekcije
```

Definisanje poruka

Izgled messages.properties datoteke

Poruke se po potrebi mogu parametrizovati, tako što se pri preuzimanju poruke prosleđuje dodatni sadržaj u poruku.

Prosleđeni sadržaj se insertuje redom po lokacijama koje se označavaju sa vitičastim zagradama { }, između kojih se navodi broj prosleđene poruke

Simple Properties Editor plug-in za Ecipse
https://marketplace.eclipse.org/content/simple-properties-editor

```
test1=Ovo je test poruka
test2=Ovo je test poruka koja sadr\u017Ei vrednosti: {0} , {1} i {2}!
```

messages.properties,
messages_sr.properties,
messages_de.properties

messages_sr.properties

```
1 #za testitranje u kontroleru
2 test1=Ovo je test poruka
3 test2=Ovo je test poruka koja sadrži vrednosti: {0} , {1} i {2}!
4 #jezik
```

Izvor poruka

MessageSource interfejs

MessageSource interfejs se u Spring radnom okviru koristi za pronalaženje poruka, pružajući podršku za njihovu parametrizaciju i interncipnalizaciju.

Korišćenjem **MessageSource** programerima je omogućen pristup tekstualnom sadržaju koje su napisan za različite jezike.

Spring sadrži dve implementacije interfejsa *ResourceBundleMessageSource* i *ReloadableResourceBundleMessageSource*.

Pristup **MessageSource** omogućen je DI mehanizmom

```
@Autowired
private MessageSource messageSource;
```

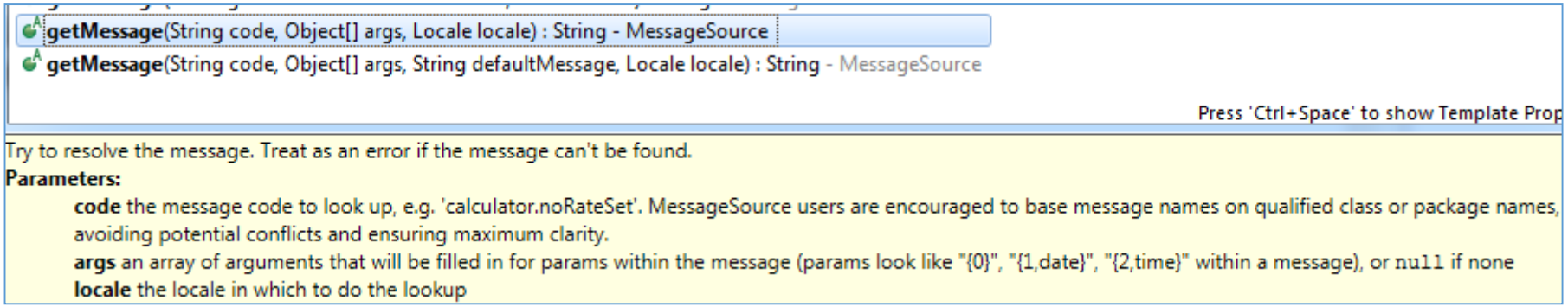
Izvor poruka

MessageSource

Metoda getMessage interfejsa **MessageSource** omogućuje pribavljanje sadržaja za navedeni jezik.

U okviru poziva metode se navode:

- ključ poruke,
- dodatni parametri poruke (ako postoje)
- lokalizacija za koju se traži poruka



Izvor poruka

MessageSource

messages_sr.properties

```
test1=Ovo je test poruka
test2=Ovo je test poruka koja sadr\u017Ei vrednosti: {0} , {1} i {2}!
```

messages.properties

```
test1=This is a test message
test2=This is a test message containing values: {0} , {1} and {2}!
```

```
messageSource.getMessage("test1",null, Locale.forLanguageTag("sr"));
messageSource.getMessage("test2",new Object[] {"Pera" , "Mika", "Žika"}, Locale.ENGLISH);
```

Ovo je test poruka
This is a test message containing values: Pera , Mika and Žika!

Izvor poruka

ResourceBundleMessageSource

Klasa ResourceBundleMessageSource predstavlja implementaciju MessageSource interfejsa i u određenoj meri predstavlja ekstenziju sandardne klase ResourceBundle iz Jave.

Spring Application Context delegira razrešenje poruka beanu čiji je naziv ***messageSource***.

Bean *messageSource* predstavlja objekat klase ResourceBundleMessageSource.

```
//definisanje bean messageSource koji Spring po defaultu koristi razrešavanje poruka
@Bean(name= {"messageSource"})
public ResourceBundleMessageSource messageSource() {
    ResourceBundleMessageSource source = new ResourceBundleMessageSource();
    //postavljanje diretorijuma koji sadrži poruke/prefiks naziva property datoteke
    source.setBasenames("messages/messages");
    //ukoliko se ne postoji poruka za ključ ispiši samo ključ
    source.setUseCodeAsDefaultMessage(true);
    source.setDefaultEncoding("UTF-8");
    //postavljanje default lokalizacije na nivou aplikacije
//  source.setDefaultLocale(Locale.forLanguageTag("sr"));
//  source.setDefaultLocale(Locale.US);
    return source;
}
```

InternacionalizacijaController i index() i ispis na konzolu

SecondConfiguration

Korišćenje različitih poruka za različite korisnike

Koristiti sesiju - mauelno upravljati lokalizacijom

Čuvanje podataka o odabranoj lokalizaciji za klijenta može se uraditi oslanjajući se na sesiju.

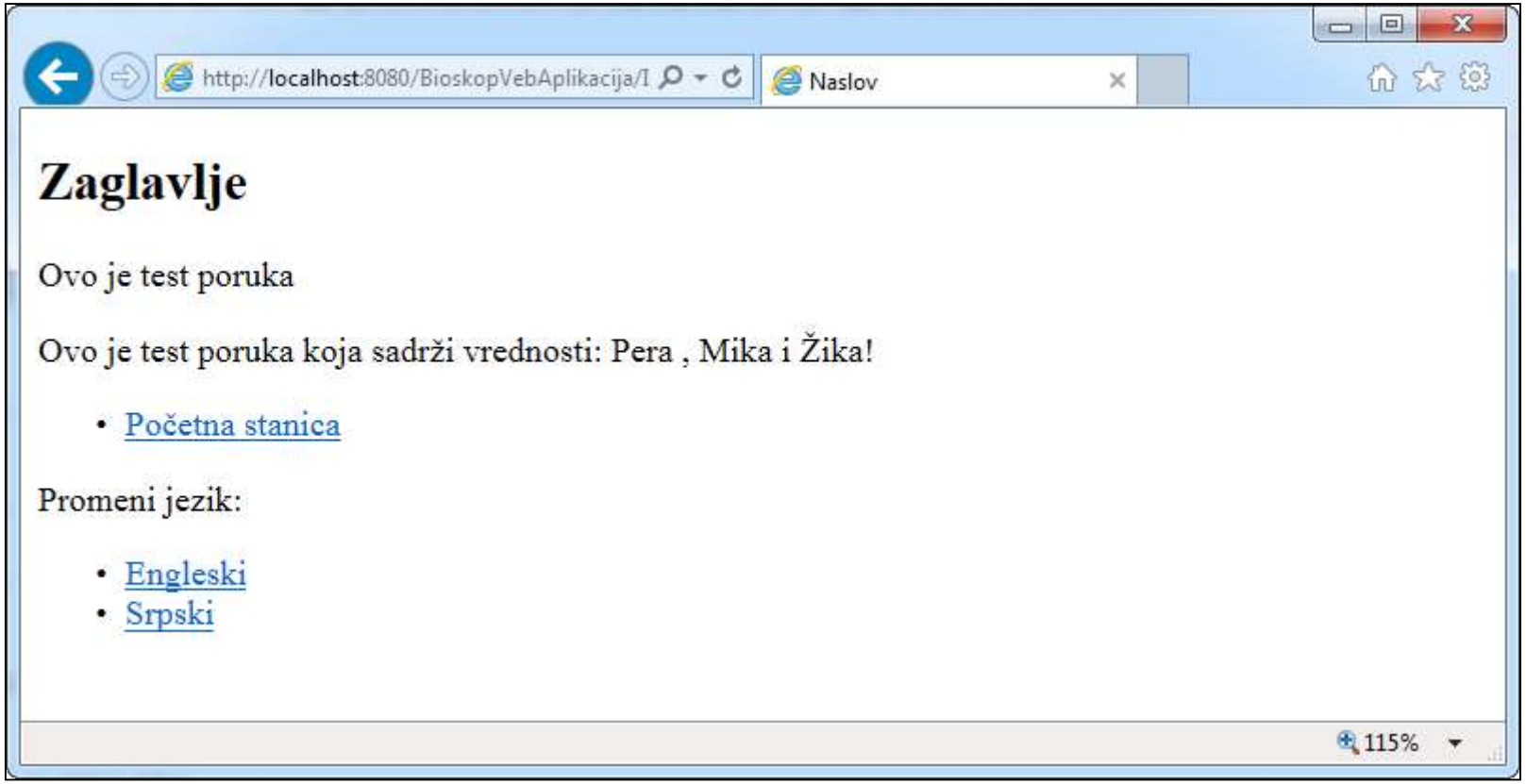
U HttpSession se može mauelno ubaciti podatak o lokalizaciji koji će se preuzeti pri definisanju teksualnog sadržaja za HTML stanice.

```
//pri inicijalizaciji sesije zadaje se lokalizacija
session.setAttribute("lokalizacija", Locale.forLanguageTag("sr"));
```

```
//u kontroleru se preuzimaju podaci o lokalizaciji
Locale lokalizacija = (Locale) session.getAttribute("lokalizacija");
...
"<title>"+messageSource.getMessage("inter.title", null, lokalizacija) + "</title>\r\n" +
```

Korišćenje različitih poruka za različite korisnike

Koristiti sesiju - mauelno upravljati lokalizacijom



InitHttpSessionListener, InternacionalizacijaController i index() i ispis u HTML

Korišćenje različitih poruka za različite korisnike

Koristiti sesiju - mauelno upravljati lokalizacijom

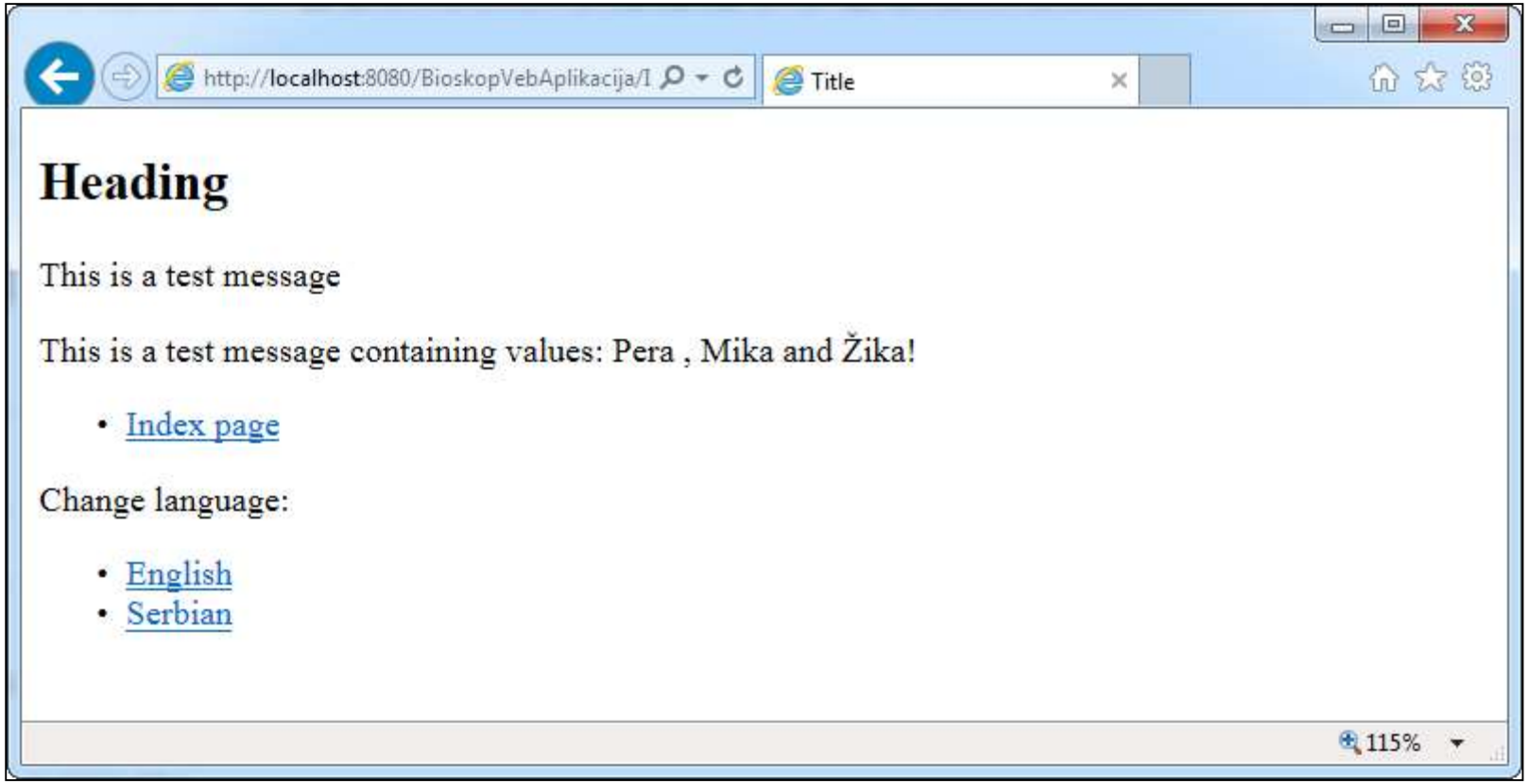
Problem nastaje usred potrebe za promenom lokalizacije koju koristi klijent.

Neophodno je kreirati metode kontrolera koje će u odnosu za dadati parametar zahteva (npr. `@RequestParam(defaultValue="sr") String jezik`) izmeniti stanje lokalizacije.

```
// GET: Internacionalizacija/PromeniJezik?jezik=en
@GetMapping("PromeniJezik")
public void promeniJezik(@RequestParam(defaultValue="sr") String jezik, HttpSession session,
HttpServletResponse response){
        Locale lokalizacija = (Locale) session.getAttribute("lokalizacija");
        if(jezik.equals("sr")) {
                lokalizacija = Locale.forLanguageTag("sr");
        } else if (jezik.equals("en")) {
                lokalizacija = Locale.ENGLISH;
        }
        session.setAttribute("lokalizacija", lokalizacija );
        response.sendRedirect(bURL+"Internacionalizacija");
}
```

Korišćenje različitih poruka za različite korisnike

Koristiti sesiju - mauelno upravljati lokalizacijom



InternacionalizacijaController i promeniJezik

Korišćenje različitih poruka za različite korisnike

Koristiti sesiju – automatski upravljati lokalizacijom

Ideja je da se upravljanjem podacima za lokalizaciju izvšrava automatski.

LocaleResolver interfejs se u Spring radnom okviru koristi za određivanje trenutne lokalizacije koja je u upotrebi za klijenta.

Postoje više implementacija inerfejsa koji određuju lokalizaciju na osnovu podataka iz HTTP zahteva koji se čuvaju u sesiji, cookie, *Accept-Language* atibutu HTTP zaglavlja.

- AcceptHeaderLocaleResolver - određuje lokalizaciju u koristeći vrednost *Accept-Language* atibut HTTP zaglavlja
- SessionLocaleResolver - određuje lokalizaciju i skladišti je u HttpSession klijenta
- CookieLocaleResolver - određuje lokalizaciju i skladišti je u cookie za klijenta

Korišćenje različitih poruka za različite korisnike

SessionLocaleResolver

Spring Application Context delegira razrešenje lokalizacije bean čiji je naziv *localeResolver.*

Bean *localeResolver* predstavlja objekat klase SessionLocaleResolver.

Neophodno je samo još definisati mehanizam za promenu lokalizacije

```
//LocaleResolver određuju lokalizaciju na osnovu podataka iz HTTP zahteva
//SessionLocaleResolver - određuje lokalizaciju i skladišti je u HttpSession klijenta
@Bean
public LocaleResolver localeResolver() {
    SessionLocaleResolver slr = new SessionLocaleResolver();
    //postavljanje default lokalizacije
    slr.setDefaultLocale(Locale.forLanguageTag("sr"));
    return slr;
}
```

Korišćenje različitih poruka za različite korisnike

LocaleChangeInterceptor

Promenu lokalizacije od stane kijenata trebalo bi programski definisati na svakoj pristupnoj tački aplikacije, što bi u suštini bilo ponavljanje istog koda ako bi ga pisali po ugledu na metodu **promeniJezik** od **InternacionalizacijaController**.

Prethodni problem rešiće se uz pomoći presretača (*interceptor*) HTTP zahteva.

Definisaće se bean localeChangeInterceptor koji će omogućiti promenu trenutne lokalizacije u odnosu na odabrani query parametar HTTP zaglavlja.

```
//Interceptor that allows for changing the current locale on every request,
//via a configurable request parameter (default parameter name: "locale").
@Bean
public LocaleChangeInterceptor localeChangeInterceptor() {
    LocaleChangeInterceptor lci = new LocaleChangeInterceptor();
    lci.setParamName("locale");
    return lci;
}
```

Korišćenje različitih poruka za različite korisnike

LocaleChangeInterceptor

Svi presretači HTTP zahteva moraju se registrovati u Spring radnom okviru.

Restracija se postiže dodavanjem presretača u InterceptorRegistry i proširivanjem @Configuration anotirane klase implementacijom *WebMvcConfigurer* interfejsa.

```
@Configuration
public class SecondConfiguration implements WebMvcConfigurer{
...

        //Restracija presretača se postiže dodavanjem presretača u InterceptorRegistry i
        //implementacijom WebMvcConfigurer interfejs
        @Override
        public void addInterceptors(InterceptorRegistry registry) {
                registry.addInterceptor(localeChangeInterceptor());
        }
}
```

SecondConfiguration

Korišćenje različitih poruka za različite korisnike

Koristiti sesiju – automatski upravljati lokalizacijom

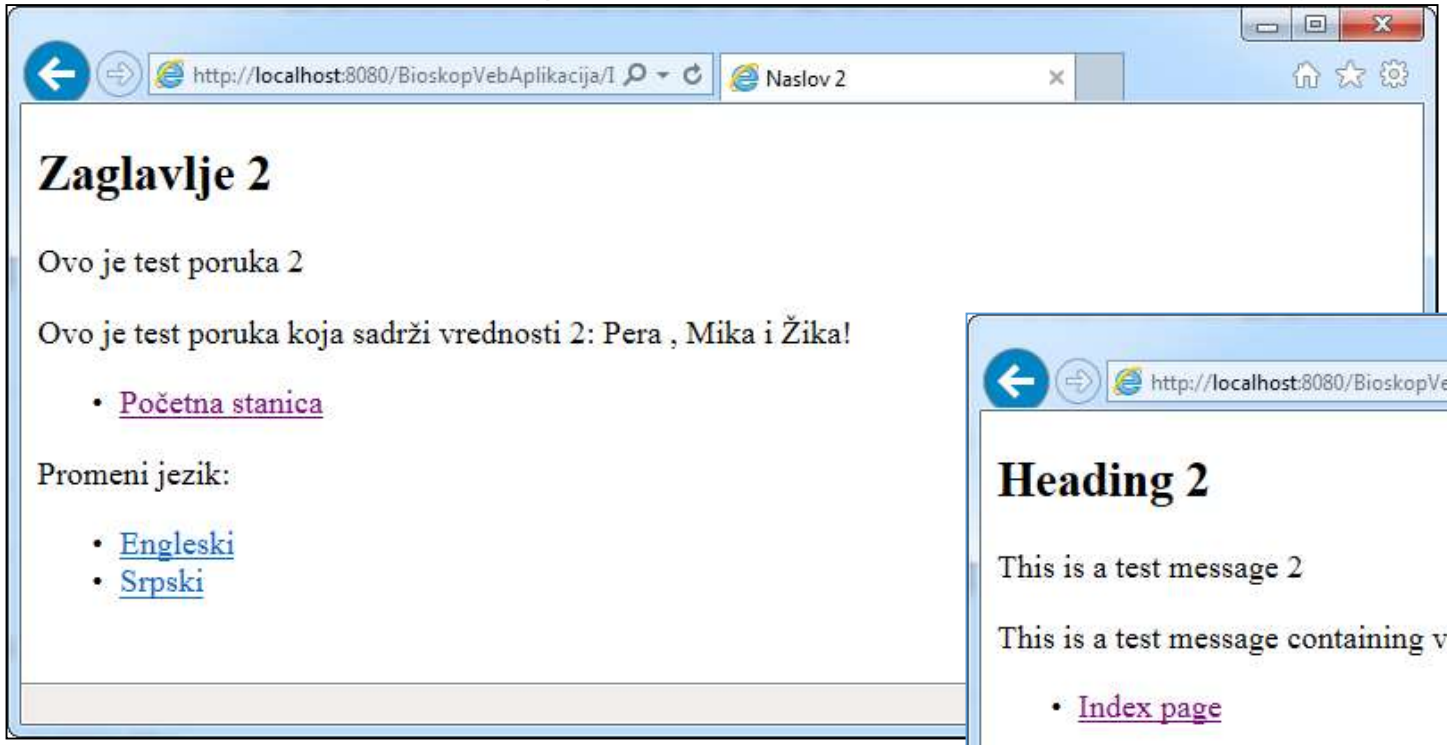
U okviru kontrolera moguće se odrediti lokalizaciju koristeći localeResolver bean i metodu resolveLocale koja kao parametar prime HTTPRequest objekat.

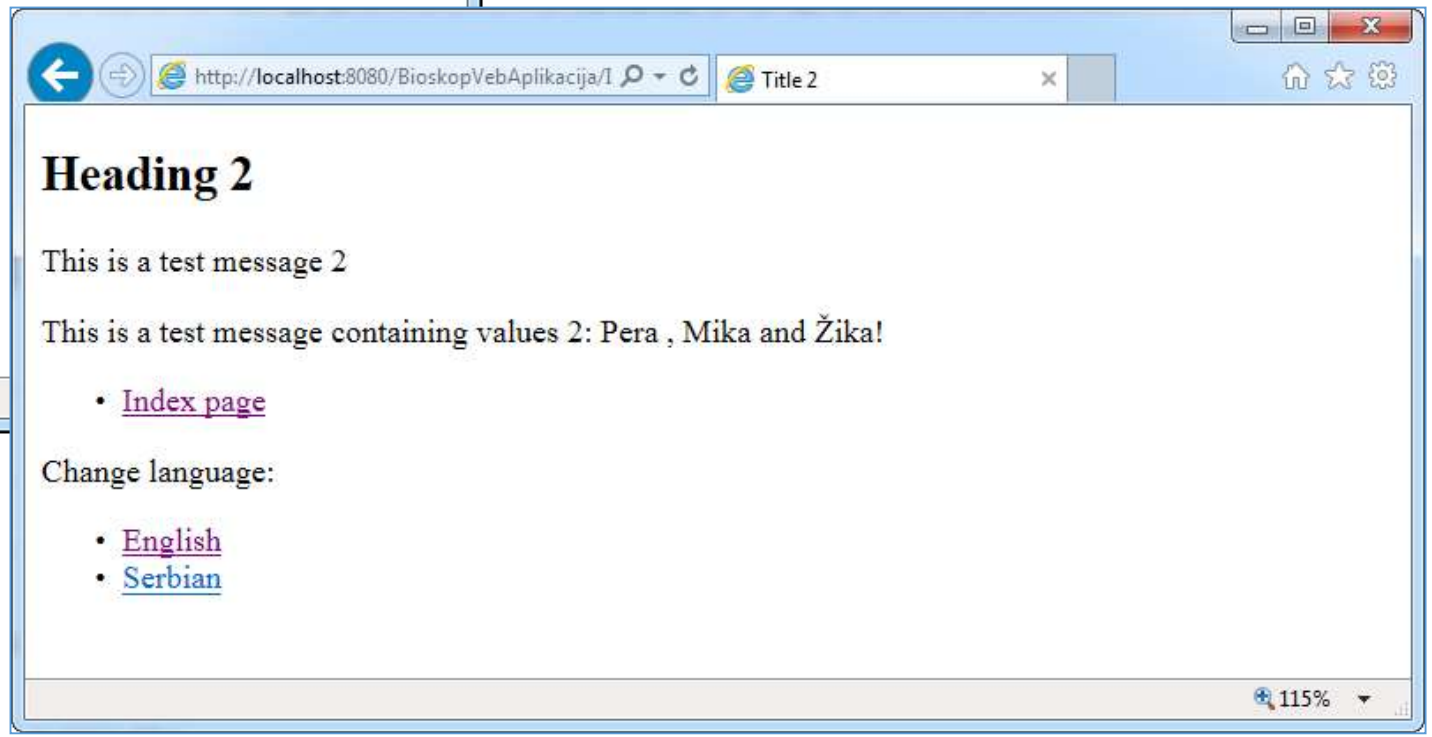
```
@Autowired
private MessageSource messageSource;
@Autowired
private LocaleResolver localeResolver;

// GET: Internacionalizacija2
@GetMapping
@ResponseBody
public String index(HttpServletRequest request) {
        Locale lokalizacija = localeResolver.resolveLocale(request);
        ...
        "<title>"+messageSource.getMessage("inter2.title", null, lokalizacija) + "</title>\r\n" +
```

Korišćenje različitih poruka za različite korisnike

Koristiti sesiju – automatski upravljati lokalizacijom

Internacionalizacija2Controller i index()





Dodatno

Koristiti sesiju – automatski upravljati lokalizacijom

Više o lokalizaciji u Spring Boot možete videti na

https://dzone.com/articles/learn-how-to-internationalize-and-localize-your-ja

https://www.baeldung.com/spring-boot-internationalization/

https://lokalise.com/blog/spring-boot-internationalization/

https://phrase.com/blog/posts/spring-boot-internationalization/