Western Norway
University of
Applied Sciences

DAT152 – Advanced Web Applications

Globalization, Internationalization, Localization, and Translation

g11n, i18n, l10n, and t9n

Goal for Today

- Understand the GILT concepts
  - Globalization (g11n)
  - Internationalization (i18n)
  - Localization (l10n)
  - Translation (t9n)
- Understand the implementation in Java

GILT – Short videos

- Concepts
- GILT
  - https://youtu.be/aZVWiHQm9JY?feature=shared
- i18n
  - https://youtu.be/CIwNjfTlmgA?feature=shared
- l10n
  - https://youtu.be/dRFSTPGY_Jk?feature=shared

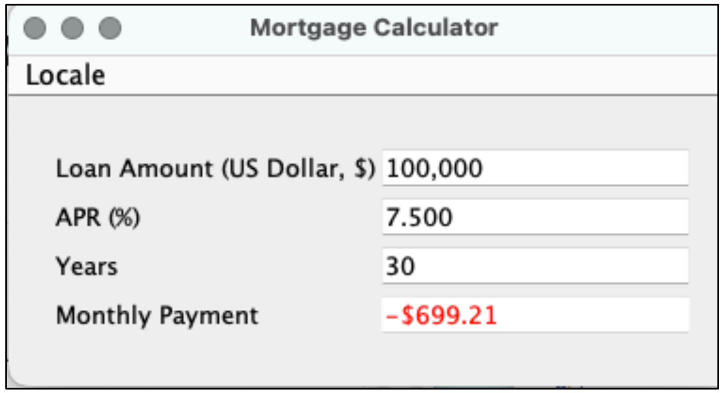
Globalization

- Your product is developed in Norway and works well in Norway
- How do you make it available globally?
- Examples
  - 1,000.00 in the US and 1.000,00 in Norwegian
  - 08/12/2022 is 12th Aug. in the US, may be interpreted as 8th Dec. in Norwegian

Many things to think about…

Globalization

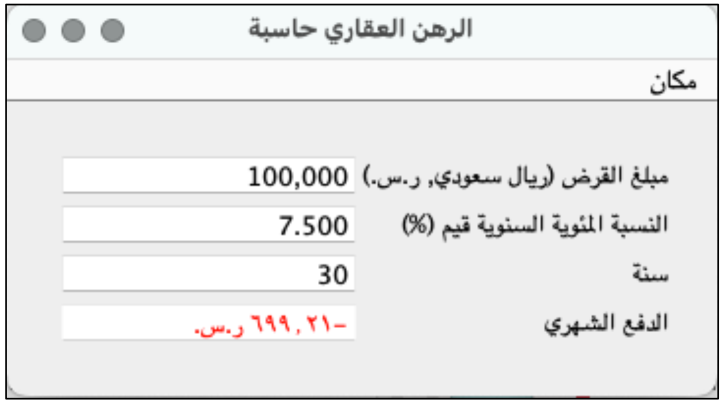
- Examples



Mortgage Calculator

Locale

Loan Amount (US Dollar, $)	100,000
APR (%)	7.500
Years	30
Monthly Payment	-$699.21

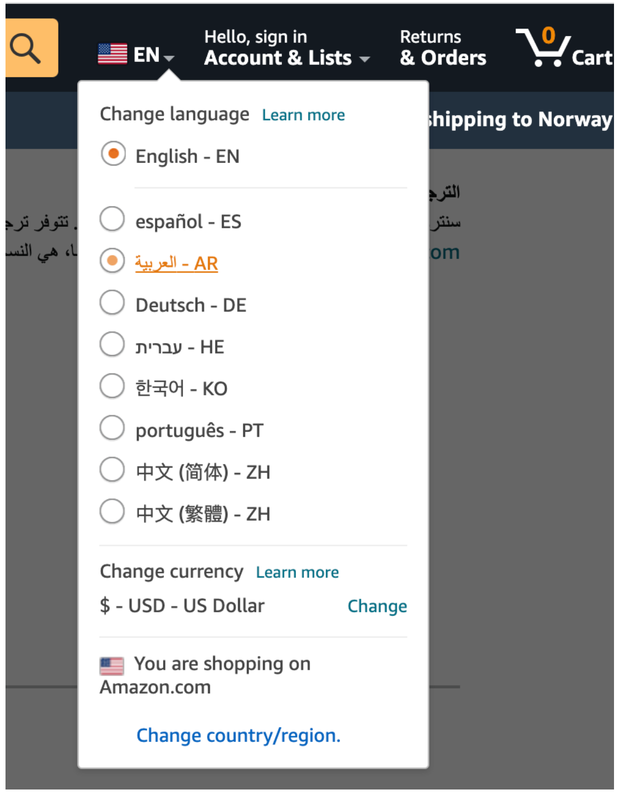


الرهن العقاري حاسبة

مكان

مبلغ القرض (ريال سعودي, ر.س.)	100,000
النسبة المئوية السنوية قيم (%)	7.500
سنة	30
الدفع الشهري	-٦٩٩,٢١ ر.س.

https://docs.oracle.com/javase/tutorial/i18n/text/bidi.html



Change language Learn more

- [✓ English - EN]
- español - ES
- العربية - AR
- Deutsch - DE
- עברית - HE
- 한국어 - KO
- português - PT
- 中文 (简体) - ZH
- 中文 (繁體) - ZH

Change currency Learn more

$ - USD - US Dollar Change

You are shopping on Amazon.com

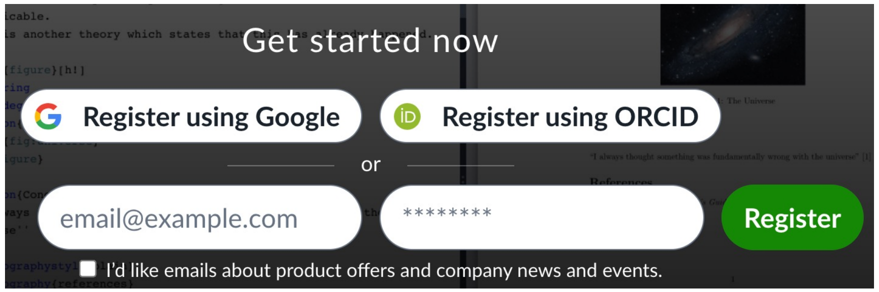
Change country/region.

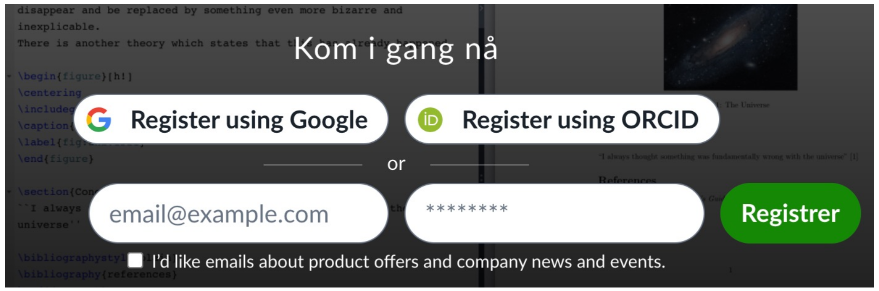
Globalization Concepts

- An application that can present information to users according to regional cultural conventions is said to be **globalized**
  - g11n = g + 11characters between g&n + n
- **Internationalization** is the process of making software portable between languages or regions
  - i18n = i + 18characters btw i&n + n
- **Localization** is the process of adapting software for specific languages or regions, called **locale**.
  - l10n = l + 10characters btw l&n + n
- **Translation** (t9n) is about expressing a word or sentence from one language to another
  - Therefore, translation is part of the process of localization.

Globalization Concepts

- A globalized application:
  - must be configurable to interact with users from different localities in culturally appropriate ways.
  - a user in one region sees error messages, output, and interface elements in the requested language. Date and time formats, as well as currencies, are presented appropriately for users in the specified region.





Globalization Concepts

- Globalization consists of two phases:
- Internationalization
  - enabling an application component for multicultural support
- and localization
  - translating and implementing a specific regional convention

Localization (l10n)

- What are the requirements for localizing the application to specific regional cultural context?
- Examples:
- What traffic colors to use in US, Japan, Norway, etc
  - US and Norway use green, yellow, red
  - Japan uses blue, yellow, and red traffic lights
- Text in a button written in specific language may become tiny or longer in another language
- Date formats can be different

Internationalization (i18n)

- How to make product function internationally from technical perspective and responsive to the long list of l10n requirements
- International software can be developed using interfaces that modify program behavior at runtime in accordance with specific cultural requirements.
  - Manipulate how data is delivered
  - Dynamically change the design from RTL language
  - Pluralize words
  - Manage time zones
  - etc

Translation (t9n)

- Translating content into a native language but also taking into consideration the context in which the content is used.
- Managed by l10n team who coordinates with professional translators or native speakers.

Checklist – Internationalization (i18n)

- Identify Culturally Dependent Data
  - Messages
  - Labels on GUI components
  - Online help
  - Sounds
  - Colors
  - Graphics
  - Icons
  - Dates
  - Times
  - Numbers
  - Currencies
  - Measurements
  - Phone numbers
  - Honorifics and personal titles
  - Postal addresses
  - Page layouts

https://docs.oracle.com/javase/tutorial/i18n/intro/checklist.html

Checklist – Internationalization (i18n)

- Isolate Translatable Text in Resource Bundles
  - status messages,
  - error messages,
  - log file entries,
  - and GUI component labels.
  - This text is included into programs that haven't been internationalized.
  - You need to locate all occurrences of included text that is displayed to end users and refactor them.

```
String buttonLabel = "OK";
// ...
JButton okButton = new JButton(buttonLabel);
```

Checklist – Internationalization (i18n)

- Deal with Compound Messages that contain variable data
  - The example code below is difficult to translate, because
    - The position of the integer in the sentence is not the same in all languages
    - the order of the sentence elements is fixed by concatenation

```
Integer fileCount;
// ...
String diskStatus = "The disk contains " + fileCount.toString() + " files";
```

NO: ?
DE: ?
ES: ?

Checklist – Internationalization (i18n)

- Format Numbers and Currencies
  - Numbers and currencies must be formatted in a local-independent manner
  - The code below is not yet internationalized

```
Double amount; TextField amountField;
// ...
String displayAmount = amount.toString();
amountField.setText(displayAmount);
```

- Format Dates and Times
  - Date and time formats differ with region and language
  - The code below is not yet internationalized

```
Date currentDate = new Date();
TextField dateField;
// ...
String dateString = currentDate.toString();
dateField.setText(dateString);
```

Localization (l10n)

- Regional cultural context like
  - Language, country, culture, appearance
- Different standards
  - Language:
    - ISO 639-1 (en, no, nn, nb, fr, cs, nl, …)
    - ISO 639-2 (eng, nor, nno, nob, fra, ces, nld, …)
  - Country:
    - ISO 3166-1 A2 (US, GB, NO, FR, ...)
    - ISO 3166-1 A3 (USA, GBR, NOR, FRA, …)
- Sometimes, we may want a more precise specification (variant)

Localization Support in Java

- Java provide support for localization through
  - **java.util.Locale** class
- Locale objects accept language, and optionally country and variant.
- Locale noLocale = new Locale("no", "NO");

Localization Support in Java

- Localization (Formatting based on Locales)
- Date and time
  - DateFormat.getDateInstance(…)
  - DateFormat.getTimeInstance(…)
- Currency
  - NumberFormat.getCurrencyInstance(…)
- Integer (+ rounding)
  - NumberFormat.getInstance(…)

Example – i18n and l10n

- A sample code illustrating i18n and l10n
- The code below is not internationalized. Why?
- Only localized to English

```
public class NotI18N {

    static public void main(String[] args) {
        System.out.println("Hello.");
        System.out.println("How are you?");
        System.out.println("Goodbye.");
    }
}
```

Source – https://docs.oracle.com/javase/tutorial/i18n/intro/before.html

Example – i18n and l10n

- Let’s refactor the code to make it internationalized
- Steps
  - Hardcoded messages must be removed
  - Language code will then be specified at run time
- Thus,
  - Same executable program can be distributed worldwide
  - No recompilation is required for localization

Example – i18n and l10n

- Create the properties Files for each language
  - English translator (save as LocaleMessages_en_EN.properties)
    - greetings = Hello
    - farewell = Goodbye
    - inquiry = How are you?
  - Norwegian translator (save as LocaleMessages_no_NO.properties)
    - greetings = Hei
    - farewell = Ha det
    - inquiry = Hvordan går det?
- Text are loaded without changing the source code…

Example – i18n and l10n

- Define the Locale or get the default locale
  - Locale locale = Locale.getDefault();
  - Local locale = new Locale("no", "NO");   //language and country codes
- Create a ResourceBundle
  - Contains locale-specific objects
  - messages = ResourceBundle.getBundle(”LocaleMessages", locale);
  - locale object specifies which property file will be loaded based on postfix "_no_NO"

Example – i18n and l10n

- Fetch the Text from the ResourceBundle
  – The property file loaded (e.g., LocaleMessages_no_NO) contain a key-value pairs
    - Values = translated text that the program will display
    - Keys must not be changed by translators

```
String language = "no";    // can be passed from environment, cmd, etc
String country = "NO";

ResourceBundle messages;
Locale currentLocale = new Locale(language, country);
messages = ResourceBundle.getBundle(”LocaleMessages", currentLocale);
System.out.println(messages.getString("greetings"));
System.out.println(messages.getString("inquiry"));
System.out.println(messages.getString("farewell"));
```

Java ResourceBundle

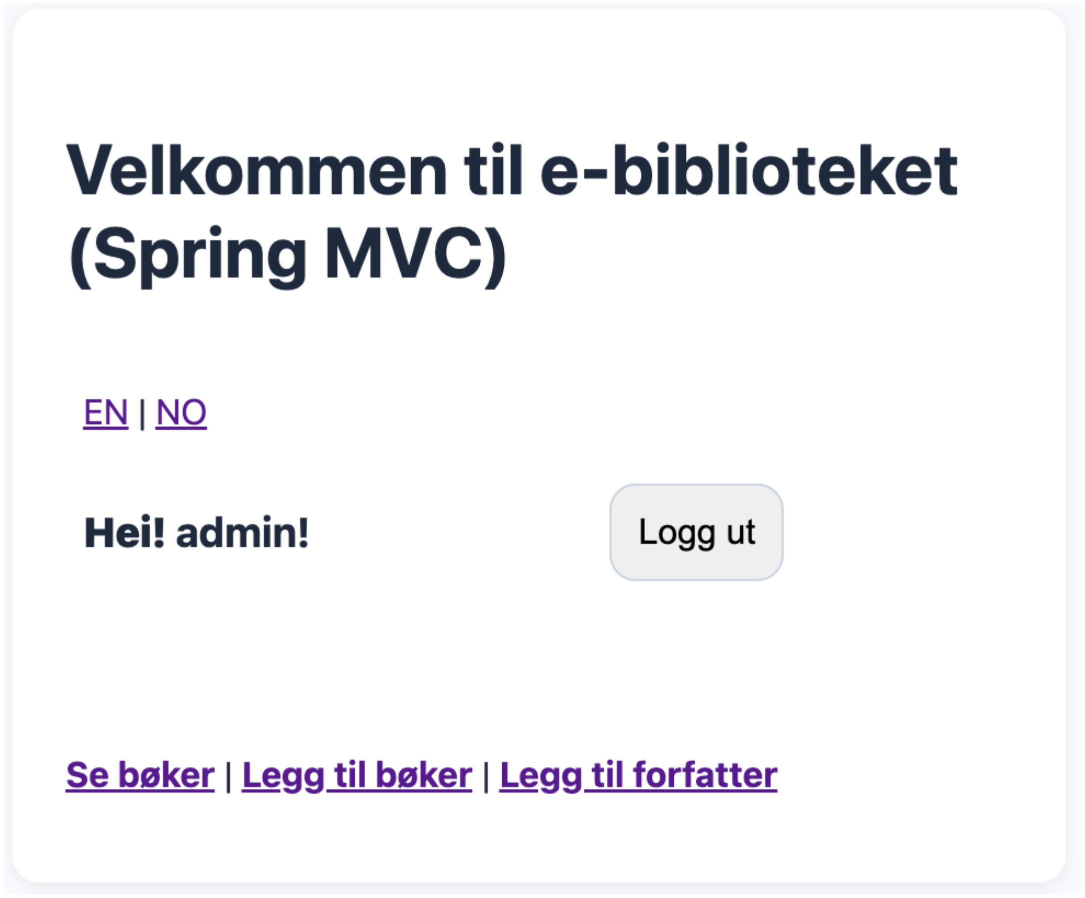
- An important class in Java i18n
- Used to separate localized content from the source code
- Variants
  - ResourceBundle,
    - Load resources in Java class objects or property files
  - ListResourceBundle
    - Load resources defined in Java class objects
  - PropertyResourceBundle
    - Used to load resources in property files

Summary

	Internationalization	Localization	Globalization
Definition	Functional in any language and content (linguistic and cultural data) separated from code/ functionality	Adaptation of products, services, and digital content to a cultural-linguistic market	The strategy of bringing a product or service to the global market, involving sales and marketing
People	Software developers, producers and authors of digital content	Translators, proofreaders, software engineers, project managers, testers, publishers	Marketing and sales personnel
Stage	Development and design of a digital product (content) or service (pre-requisite for localisation)	Translation and adaptation of text, user interface, and cultural conventions	Bringing to the market the internationalised and localised product or service

Anastasiou, D., & Schäler, R. (2010). Translating vital information: Localisation, internationalisation, and globalisation. Synthèses journal, 3(11), 11-25.

Spring MVC – i18n support



Velkommen til e-biblioteket (Spring MVC)

EN | NO

Hei! admin!

Se bøker | Legg til bøker | Legg til forfatter

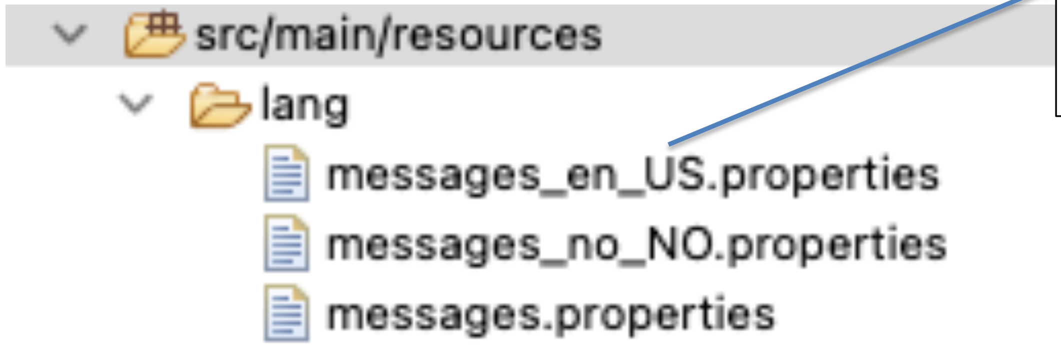
Key components and configuration

There are a few configuration needed to internationalize spring boot applications

- MessageSource Interface (Localization context)
  - Add translation files
  - Configure the resourcebundle
- LocaleResolvers
  - Configure the type of locale resolver (see next slide)
- LocaleChangeInterceptor
  - Configure the interceptor for the request when locale changes
- Controller
  - Handles the redirection to the referrer page after locale request interception

Localization Context

- Create the resource bundles according to the localization context
- Include a base resource bundle for fallback



src/main/resources
lang
messages_en_US.properties
messages_no_NO.properties
messages.properties

```
greet=Welcome to e-Library (Spring MVC)
home=Home
view=View Books
add=Add Books
author=Add Author
```

LocaleResolver

- FixedLocaleResolver
  - always resolves the locale to a singular fixed language mentioned in the project properties.
- AcceptHeaderLocaleResolver
  - resolves the locale using an “Accept-Language” HTTP header retrieved from an HTTP request.
- SessionLocaleResolver
  - resolves the locale and stores it in the HttpSession of the user
- CookieLocaleResolver
  - resolves the locale and stores it in a cookie on the client’s browser storage.

LocaleResolver

- Configure a locale resolver. Here, we are using CookieLocaleResolver
- Spring creates a locale cookie on the client's browser with the name:
  - org.springframework.web.servlet.i18n.CookieLocaleResolver.LOCALE

```
@Bean
public LocaleResolver localeResolver() {
    CookieLocaleResolver localeResolver = new CookieLocaleResolver();
    localeResolver.setDefaultLocale(Locale.ENGLISH);
    localeResolver.setDefaultTimeZone(TimeZone.getTimeZone("UTC"));

    return localeResolver;
}
```

LocaleChangeInterceptor

- Create an interceptor bean with a parameter that should be observed on the HTTP request. Here, we are using “locale”.
- Register the interceptor

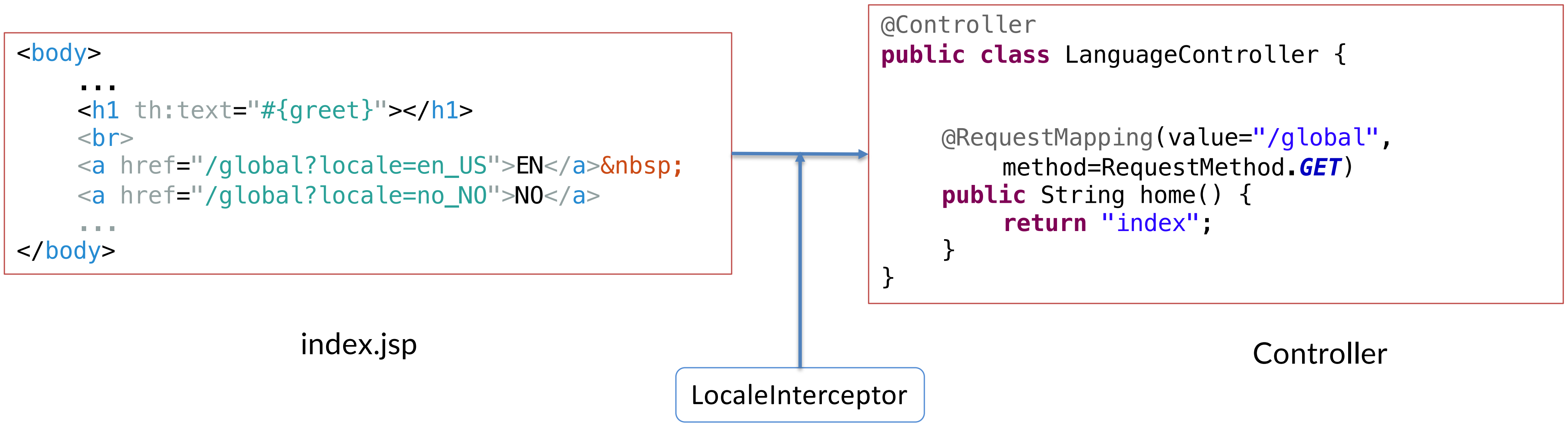
```java
@Bean
public LocaleChangeInterceptor localeChangeInterceptr() {
    LocaleChangeInterceptor localeChgInterceptor = new
    LocaleChangeInterceptor();
    // interceptor intercept request with param=locale
    localeChgInterceptor.setParamName("locale");

    return localeChgInterceptor;
}
```

```java
@Override
public void addInterceptors(InterceptorRegistry registry) {
    registry.addInterceptor(localeChangeInterceptr());
}
```

Controller

- Lastly, we need a controller to manage the redirection to the locale request referrer.



```
<body>
    ...
    <h1 th:text="#{greet}"></h1>
    <br>
    <a href="/global?locale=en_US">EN</a>&nbsp;
    <a href="/global?locale=no_NO">NO</a>
    ...
</body>
```

index.jsp

```
@Controller
public class LanguageController {


    @RequestMapping(value="/global",
        method=RequestMethod.GET)
    public String home() {
        return "index";
    }
}
```

Controller

Demo

Next – Universal Design

- Self-Study (No Lecture!): Friday 17.10.2024
- Group or individual
- Obligatory
- 5 mins recorded presentation and uploaded on Canvas
- Preparatory material provided on Canvas