# Internationalization and Location

|  |  |
|---|---|
| Created | @Oct 6, 2020 5:34 PM |
| Tags | |
| Updated | @Oct 6, 2020 5:36 PM |

Imagine that you have brought a nice imported TV from Japan that is equipped with a wonderful set of features. You have opened the box and found an instruction manual that is written in the Japanese. Will you be comfortable in operating the TV with the given set of instructions?

Same is the case with any software application. Consider, you want to cater your online retail website to the people across the world and you have not presented the website interface to the user in their local language. This definitely hampers the user experience and in-turn has a lot of impact on your global market revenue.

This is where Internationalization (**i18n**) comes into picture.

*Internationalization* is the process of ensuring that your website accommodates multiple languages and cultural conventions to make the creation of localized sites possible.

**Spring Boot provides a lot of built in features to support Internationalization through ResourceBundles, LocaleResolvers and Interceptors.**

- Default locale file should be named as **messages.properties**
- Other locale files will be named as **messages_<<*LocaleName*>>.properties** ( where LocaleName is any of '**fr**' for French, '**de**' for German etc….)

```
@Configuration
public class InternationalizationConfig implements WebMvcConfigurer {

/**
The default implementation is AcceptHeaderLocaleResolver, simply using the request's locale provided by the respective HTTP hea
In our example, we are using SessionLocaleResolver to resolve the current user's locale.
*/
    @Bean
    public LocaleResolver localeResolver() {
        SessionLocaleResolver localeResolver = new SessionLocaleResolver();
        localeResolver.setDefaultLocale(Locale.US);
        return localeResolver;
    }


/**
Interceptor allows for changing the current locale on every request, via a configurable request parameter (default parameter na
For the sake of simplicity, in our example, we used the parameter name as "lang".
*/
    @Bean
    public LocaleChangeInterceptor localeChangeInterceptor() {
        LocaleChangeInterceptor localeChangeInterceptor = new LocaleChangeInterceptor();
        localeChangeInterceptor.setParamName("lang");
        return localeChangeInterceptor;
    }

/**
We need to register this interceptor with Spring Boot Registry.
*/
    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(localeChangeInterceptor());
    }
}
```