Notatki - Internationalization i Localization

Spis treści

Internationalization i Localization

Na początek pogadamy sobie o lokalizacji. Dużo aplikacji jest używanych przez użytkowników z różnych krajów, a co za tym idzie, aplikacje muszą funkcjonować w różnych językach. Oprócz tego, że aplikacje powinny funkcjonować z możliwością wyboru języka, to często konieczne jest zapisywanie dat lub pieniędzy w określonych formatach. Mogą to być również przecinki lub kropki w liczbach.

Przykład, nad którym warto się chwilę zastanowić. Data `6/1/10` dla nas oznacza `6 styczeń 2010`, w US natomiast będzie oznaczała `1 czerwiec 2010`. Jak już o tym wiemy, to pisząc program należy takie sytuacje wspierać i obsługiwać 😉.

Wyjaśnijmy sobie dwa terminy:

- **Internationalization** - proces projektowania programu w taki sposób, aby można było ten program dostosowywać do różnych języków lub regionów bez dokonywania zmian w kodzie. Oznacza to, że aplikacja, którą piszemy może być konfigurowalna przez pliki z "jakimiś ustawieniami". Efektem jest sytuacja, w której w zależności od ustawień możemy stosować inne formaty dat dla innych preferencji użytkownika. W zasadzie, to nie musimy wspierać wielu języków jednocześnie, wystarczy wspierać więcej niż jeden język. Termin ten odnosi się do tego, że jesteś przygotowany na obsługę wielu języków w swoim oprogramowaniu.
- **Localization** - proces dostosowywania oprogramowania do określonego regionu lub języka. W wyniku takiego procesu, aplikacja faktycznie wspiera wiele lokalizacji (dalej będziemy używać określenia `locale`). Java określa lokalizację (`locale`) jako konkretny region geograficzny/polityczny/kulturowy. `Locale` można rozumieć jako para język oraz kraj. Możliwa jest sytuacja gdzie kilka krajów używa tego samego języka, ale w zależności od kraju, różnią się pewne aspekty formatowania. Mówiąc lokalizacja, mamy też na myśli, że aplikacja wspiera słowa/zdania/stringi w różnych językach. Jednocześnie zawiera w tym się kwestia formatowania dat i liczb w formacie zgodnym z danym `locale`. Za każdym razem jak będziemy dodawać do aplikacji nowy język, który ma być wspierany, musimy mieć wsparcie dla wszystkich istniejących tekstów/formatów wiążących się z danym językiem.

Jedno i drugie słowo jest dosyć długie, a jak już może udało Ci się zauważyć, programersy lubią skracać słowa/zdania/czynności. Dlatego też, powstały 2 skróty dla powyższych słów:

- **i18n** - Internationalization
- **l10n** - Localization

Liczba po drodze odnosi się do liczby znaków pomiędzy znakami skrajnymi.

Przechodząc do konkretów.

Podstawy Locale

Jak zdefiniować default Locale

```
Locale defaultLocale = Locale.getDefault();
System.out.println(defaultLocale);
```

Na ekranie zostanie wydrukowane `pl_PL`, gdzie:

- **pl** - oznacza, że na komputerze jest ustawiony język polski
- **PL** - oznacza, że ustawiony kraj na komputerze to Polska

Czyli pierwszy fragment przed podłogą oznacza język, a drugi po podłodze oznacza kraj. Kraj nie jest konieczny, Locale może równie dobrze wyglądać tak `pl`.

A jak można samemu określić Locale

```
System.out.println(Locale.ENGLISH); // en
System.out.println(Locale.UK); // en_GB
```

Albo przez konstruktor:

```
System.out.println(new Locale("en")); // en
System.out.println(new Locale("en", "PL")); // en_PL - no tak też można
```

Default można też zmienić na pojedyncze uruchomienie programu, przy ponownym uruchomieniu zostaną wczytane poprzednie ustawienia.

Powyższy przykład będzie działał poprawnie, gdy korzystamy z wersji Java mniejszej niż 19. Od Java 19 zaleca się wykorzystanie metody `Locale.of()`.