next>>

Internationalization and Localization

Internationalization is also abbreviated as I18N because there are total 18 characters between the first letter 'I' and the last letter 'N'. Internationalization is a mechanism to create such an application that can be adapted to different languages and regions. Internationalization is one of the powerful concept of java if you are developing an application and want to display messages, currencies, date, time etc. according to the specific region or language.

Localization is also abbreviated as I10N because there are total 10 characters between the first letter 'L' and last letter 'N'. Localization is the mechanism to create such an application that can be adapted to a specific language and region by adding locale-specific text and component.

Do You Know ?

- ***What is the use of Locale class ?***
- ***How can we globalize the messages (or) What is the use of ResourceBundle class?***
- ***How can we internationalize the date, time, number, currency and measurements?***

Understanding the culturally dependent data before starting internationalization

Before starting the internationalization, Let's first understand what are the informations that differ from one region to another. There is the list of culturally dependent data:

- Messages
- Dates
- Times
- Numbers
- Currencies
- Measurements
- Phone Numbers
- Postal Addresses
- Labels on GUI components etc.

Importance of Locale class in Internationalization

An object of Locale class represents a geographical or cultural region. This object can be used to get the locale specific information such as country name, language, varient etc.

Fields of Locale class

There are fields of Locale class:

1. public static final Locale ENGLISH
2. public static final Locale FRENCH
3. public static final Locale GERMAN
4. public static final Locale ITALIAN
5. public static final Locale JAPANESE
6. public static final Locale KOREAN
7. public static final Locale CHINESE

8. public static final Locale SIMPLIFIED_CHINESE
9. public static final Locale TRADITIONAL_CHINESE
10. public static final Locale FRANCE
11. public static final Locale GERMANY
12. public static final Locale ITALY
13. public static final Locale JAPAN
14. public static final Locale KOREA
15. public static final Locale CHINA
16. public static final Locale PRC
17. public static final Locale TAIWAN
18. public static final Locale UK
19. public static final Locale US
20. public static final Locale CANADA
21. public static final Locale CANADA_FRENCH
22. public static final Locale ROOT

Constructors of Locale class

There are three constructors of Locale class.They are as follows:

1. Locale(String language)
2. Locale(String language, String country)
3. Locale(String language, String country, String varient)

Commonly used methods of Locale class

There are given commonly used methods of Locale class.

1. **public static Locale getDefault()** it returns the instance of current Locale
2. **public static Locale[] getAvailableLocales()** it returns an array of available locales.
3. **public String getDisplayCountry()** it returns the country name of this locale object.
4. **public String getDisplayLanguage()** it returns the language name of this locale object.
5. **public String getDisplayVarient()** it returns the varient code for this locale object.
6. **public String getISO3Country()** it returns the three letter abbreviation for the current locale's country.
7. **public String getISO3Language()** it returns the three letter abbreviation for the current locale's language.

Example of Local class that prints the informations of the default locale

In this example, we are displaying the informations of the default locale. If you want to get the informations about any specific locale, comment the first line statement and uncomment the second line statement in the main method.

```
import java.util.*;
public class LocaleExample {
public static void main(String[] args) {
Locale locale=Locale.getDefault();
//Locale locale=new Locale("fr","fr");//for the specific locale

System.out.println(locale.getDisplayCountry());
System.out.println(locale.getDisplayLanguage());
System.out.println(locale.getDisplayName());
System.out.println(locale.getISO3Country());
System.out.println(locale.getISO3Language());
System.out.println(locale.getLanguage());
System.out.println(locale.getCountry());

}
}
```

download this example

```
Output:United States
       English
       English (United States)
```

```
USA
eng
en
US
```

Example of Local class that prints english in different languages

In this example, we are displaying english language in different language. Let's see how english is written in french and spanish languages.

```
import java.util.*;
public class LocaleExample2 {
        public static void main(String[] args) {
                Locale enLocale = new Locale("en", "US");
                Locale frLocale = new Locale("fr", "FR");
                Locale esLocale = new Locale("es", "ES");
                System.out.println("English language name (default): " +
                                enLocale.getDisplayLanguage());

                System.out.println("English language name in French: " +
                                enLocale.getDisplayLanguage(frLocale));
                System.out.println("English language name in spanish: " +
                enLocale.getDisplayLanguage(esLocale));
        }

}
```

```
Output:English language name (default): English
       English language name in French: anglais
       English language name in spanish: ingl?s
```

Example of Local class that print display language of many locales

In this example, we are displaying the display lanuage of many locales.

```
import java.util.*;
public class LocaleEx {
public static void main(String[] args) {
Locale[] locales = { new Locale("en", "US"),
 new Locale("es", "ES"), new Locale("it", "IT") };

for (int i=0; i< locales.length; i++) {
 String displayLanguage = locales[i].getDisplayLanguage(locales[i]);
 System.out.println(locales[i].toString() + ": " + displayLanguage);
}
}

}
```

```
Output:en_US: English
       es_ES: espa?ol
       it_IT: italiano
```

What we will learn in Internationalization Tutorial ?

- ResourceBundle class
- I18N with Date
- I18N with Time
- I18N with Number
- I18N with Currency
- I18N with Measurements