Date And Time

Java Locale

You will need to work with java locale api when you want to display numbers, dates, and time in a user-friendly way that conforms to the language and cultural expectations of your customers. In java, **java.util.Locale** class represents a specific language and region of the world.

If a class varies its behavior according to Locale, it is said to be locale-sensitive. For example, the **NumberFormat** and **DateFormat** classes are locale-sensitive; the format of the number and date, it returns depends on the Locale.

Create Locale Instance

You can create java locale instances in following ways:

Static Locale objects

This one is easiest and uses predefined constants in **Locale** class.

```
Locale usLocale = Locale.US;

long number = 123456789L;

NumberFormat nf = NumberFormat.getInstance(usLocale);

System.out.println( nf.format(number) );     //123,456,789

Date now = new Date();

DateFormat df = DateFormat.getDateTimeInstance(DateFormat.LONG, DateFormat.LONG, usLocale);

System.out.println( df.format(now) );    //July 19, 2016 12:43:12 PM IST
```

Please note that when locale is build this way then the region portion of the Locale is undefined. So below both statements are essentially equal:

```
//Region is missing in both cases

Locale usLocale = Locale.US;       //1

Locale usLocale = new Locale.Builder().setLanguage("en").build();       //2
```

Locale constructor

There are three constructors available in the Locale class:

Locale(String language)

Date And Time

Locale(String language, String country)

Locale(String language, String country, String variant)

```
Locale usLocale = new Locale("en");
//Locale usLocale = new Locale("en", "US");
long number = 123456789L;
NumberFormat nf = NumberFormat.getInstance(usLocale);
System.out.println( nf.format(number) );    //123,456,789
Date now = new Date();
DateFormat df = DateFormat.getDateTimeInstance(DateFormat.LONG, DateFormat.LONG, usLocale);
System.out.println( df.format(now) );    //July 19, 2016 12:43:12 PM IST
```

Methods:

getDisplayCountry() : java.util.Locale.getDisplayCountry() return the country to which locale belongs to.

Syntax :

public final String getDisplayCountry()

getDefault() : java.util.Locale.getDefault() return the current – default value for the locale as per the JVM instance.

Syntax :

public static Locale getDefault()

Return :

current - default value for the locale as per the JVM instance.

getCountry() : java.util.Locale.getCountry() return the country for the locale which could be empty or ISO 3166 2-letter code.

Syntax :

public String getCountry()

equals(Object locale2) : java.util.Locale.equals(Object locale2) checks whether two locales are equal or not.

Syntax :

public boolean equals(Object locale2)

Parameters :

locale2 : another locale to be compare with.

Return :

returns true if two locales are equal, else false.

clone() : java.util.Locale.clone() creates clone of the locale.

Syntax :

public Object clone()

Return :

clone of this instance

getAvailableLocales() : java.util.Locale.getAvailableLocales() returns array of all the installed locales.

Syntax :

public static Locale[] getAvailableLocales()

Return :

array of installed locales.

getDisplayLanguage() : java.util.Locale.getDisplayLanguage() returns the language with the locale.

Syntax :

public final String getDisplayLanguage()

getDisplayLanguage(Locale in) : java.util.Locale.getDisplayLanguage(Locale in) returns the language localised according to “in” Locale if possible.

Syntax :

public String getDisplayLanguage(Locale in)

Parameters :

in : the instance local

Exception :

NullPointerException : if "in" is null.

Date And Time

getDisplayName() : java.util.Locale.getDisplayName() displays name of the Locale

Syntax :

public final String getDisplayName()

getDisplayLanguage(Locale in) : java.util.Locale.getDisplayLanguage(Locale in) returns the language of “in” locale.

Syntax :

public final String getDisplayLanguage()

Parameters :

in : the instance local

getISO3Country() : java.util.Locale.getISO3Country() displays 3-letter abbreviation of Locale country.

Syntax :

public String getISO3Country()

Date And Time

Java Date:

Introduction

The java.util.Date class represents a specific instant in time, with millisecond precision.

Class constructors

Date()

This constructor allocates a Date object and initializes it so that it represents the time at which it was allocated, measured to the nearest millisecond.

Date(long date)

This constructor allocates a Date object and initializes it to represent the specified number of milliseconds since the standard base time known as "the epoch", namely January 1, 1970, 00:00:00 GMT.

Class methods

boolean after(Date when)

This method tests if this date is after the specified date.

boolean before(Date when)

This method tests if this date is before the specified date.

Object clone()

This method return a copy of this object.

int compareTo(Date anotherDate)

This method compares two Dates for ordering.

boolean equals(Object obj)

This method compares two dates for equality.

long getTime()

This method returns the number of milliseconds since January 1, 1970, 00:00:00 GMT represented by this Date object.

void setTime(long time)

This method sets this Date object to represent a point in time that is time milliseconds after January 1, 1970 00:00:00 GMT.

String toString()

This method converts this Date object to a String of the form.

Date And Time

Java Date Format

DateFormatTest.java

```java
package datereflection;

import java.text.DateFormat;
import java.util.Date;

public class DateFormatTest {
    public static void main( String[] args ) {

        Date currentDate = new Date( );
        System.out.println( "Date: " + currentDate );

        String dateToStr = DateFormat.getInstance( ).format( currentDate );
        System.out.println( "Date: getInstance(): " + dateToStr );

        dateToStr = DateFormat.getDateInstance( ).format( currentDate );
        System.out.println( "Date: getDateInstance(): " + dateToStr );

        dateToStr = DateFormat.getTimeInstance( ).format( currentDate );
        System.out.println( "Date: getTimeInstance(): " + dateToStr );

        dateToStr = DateFormat.getDateTimeInstance( ).format( currentDate );
        System.out.println( "Date: getDateTimeInstance(): " + dateToStr );

    }
}
```

Console | Markers | Properties | Servers | Data Source Explorer | Snippets | Progress | Er

```
<terminated> DateFormatTest [Java Application] C:\Program Files\Java\jdk1.7.0_79\bin\javaw.exe (Sep 17, 2015, 2:04:59
Date: Thu Sep 17 14:04:59 NPT 2015
Date: getInstance(): 9/17/15 2:04 PM
Date: getDateInstance(): Sep 17, 2015
Date: getTimeInstance(): 2:04:59 PM
Date: getDateTimeInstance(): Sep 17, 2015 2:04:59 PM
```

Date And Time

Simple Date Format:

SimpleDateFormatTest.java

```java
package datereflection;

import java.text.SimpleDateFormat;
import java.util.Date;

public class SimpleDateFormatTest {
    public static void main( String[] args ) {

        Date date = new Date( );

        SimpleDateFormat formatter = new SimpleDateFormat( "MM/dd/yyyy" );
        String strDate = formatter.format( date );
        System.out.println( "DateFormat: MM/dd/yyyy : " + strDate );

        formatter = new SimpleDateFormat( "dd-M-yyyy hh:mm:ss" );
        strDate = formatter.format( date );
        System.out.println( "DateFormat: dd-M-yyyy hh:mm:ss : " + strDate );

        formatter = new SimpleDateFormat( "dd MMMM yyyy" );
        strDate = formatter.format( date );
        System.out.println( "DateFormat: dd MMMM yyyy : " + strDate );
    }
}
```

Console | Markers | Properties | Servers | Data Source Explorer | Snippets | Progress | Error Lo

```
<terminated> SimpleDateFormatTest [Java Application] C:\Program Files\Java\jdk1.7.0_79\bin\javaw.exe (Sep 17, 2015, 2:08:48
DateFormat: MM/dd/yyyy : 09/17/2015
DateFormat: dd-M-yyyy hh:mm:ss : 17-9-2015 02:08:48
DateFormat: dd MMMM yyyy : 17 September 2015
```

DateStrToDateObject.java

```java
package datereflection;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class DateStrToDateObject {
    public static void main( String[] args ) {

        try {

            SimpleDateFormat formatter = new SimpleDateFormat( "dd/MM/yyyy" );
            Date date = formatter.parse( "31/03/2015" );

            System.out.println( "Date is: " + date );

        } catch ( ParseException e ) {
            e.printStackTrace( );
        }
    }
}
```

Date And Time

Joda Time Library:

JodaTimeExample.java

```java
package datereflection;

import java.util.Date;

import org.joda.time.DateTime;

public class JodaTimeExample {
    public static void main( String[] args ) {

        DateTime dateTime = new DateTime( );

        Date date = dateTime.toDate( );

        System.out.println( date );
    }
}
```

Console | Markers | Properties | Servers | Data Source Explorer

```
<terminated> JodaTimeExample [Java Application] C:\Program Files\Java\jdk1.7.0_79
Thu Sep 17 15:55:36 NPT 2015
```

Date And Time

JodaTimeExamples.java

```java
package datereflection;

import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import org.joda.time.format.DateTimeFormatter;

public class JodaTimeExamples {
    public static void main( String[] args ) {

        DateTime dt = new DateTime( ); // Joda Date
        System.out.println( "Date:" + dt.toDate( ) ); // Java Date

        int month = dt.getMonthOfYear( );
        System.out.println( "MonthOfYear: " + month );

        DateTime.Property pDoW = dt.dayOfWeek( );// Monday:1 to Sunday:7
        System.out.println( "dayOfWeek: " + pDoW.getAsText( ) ); // print:Monday/Tuesday

        System.out.println( "getDayOfMonth: " + dt.getDayOfMonth( ) );
        int maxDay = dt.dayOfMonth( ).getMaximumValue( );
        System.out.println( "Last day of this month: " + maxDay + " day" );

        boolean leapYear = dt.yearOfEra( ).isLeap( );
        System.out.println( "Leap Year: " + leapYear );

        DateTime datePlus20 = dt.plusDays( 20 );
        DateTimeFormatter formattedDate = DateTimeFormat.forPattern( "dd/MM/yyyy" );
        System.out.println( dt.toString( formattedDate ) + " + 20 day = " + datePlus20.toString( formattedDate ) );

    }
}
```

Console | Markers | Properties | Servers | Data Source Explorer | Snippets | Progress | Error Log

```
<terminated> JodaTimeExamples [Java Application] C:\Program Files\Java\jdk1.7.0_79\bin\javaw.exe (Sep 17, 2015, 4:19:38 PM)
Date:Thu Sep 17 16:19:38 NPT 2015
MonthOfYear: 9
dayOfWeek: Thursday
getDayOfMonth: 17
Last day of this month: 30day
Leap Yearfalse
17/09/2015 + 20 day = 07/10/2015
```

Date And Time

Java Reflection API

Java Reflection is a process of examining or modifying the run time behavior of a class at run time.

The java.lang.Class class provides many methods that can be used to get metadata, examine and change the run time behavior of a class.

The java.lang and java.lang.reflect packages provide classes for java reflection.

Where it is used

The Reflection API is mainly used in:

- IDE (Integrated Development Environment) e.g. Eclipse, MyEclipse, NetBeans etc.
- Debugger
- Test Tools etc.

java.lang.Class class

The **java.lang.Class** class performs mainly two tasks:

- provides methods to get the metadata of a class at run time.
- provides methods to examine and change the run time behavior of a class.

Commonly used methods of Class class:

public String getName() returns the class name

public static Class forName(String className)throws ClassNotFoundException loads the class and returns the reference of Class class

public Object newInstance() throws InstantiationException,IllegalAccessException creates new instance

public boolean isInterface() checks if it is interface

public boolean isArray() checks if it is array.

public boolean isPrimitive() checks if it is primitive

public Class getSuperclass() returns the superclass class reference.

public Field[] getDeclaredFields() throws SecurityException returns the total number of fields of this class.

public Method[] getDeclaredMethods() throws SecurityException returns the total number of methods of this class.

public Constructor[] getDeclaredConstructors() throws SecurityException returns the total number of constructors of this class.

public Method getDeclaredMethod(String name,Class[] parameterTypes) throws NoSuchMethodException,SecurityException returns the method class instance.