**New Date-Time API in Java 8**

New date-time API is introduced in Java 8 to overcome the following drawbacks of old date-time API :

Not thread safe : Unlike old java.util.Date which is not thread safe the new date-time API is immutable and doesn’t have setter methods.

Less operations : In old API there are only few date operations but the new API provides us with many date operations.

Java 8 under the package java.time introduced a new date-time API, most important classes among them are :

Local : Simplified date-time API with no complexity of timezone handling.

Zoned : Specialized date-time API to deal with various timezones.

LocalDate/LocatTime and LocalDateTime API : Use it when time zones are NOT required.

Java

|  |
| --- |
| // Java code for LocalDate  // / LocalTime Function  import java.time.\*;  import java.time.format.DateTimeFormatter;    public class Date {    public static void LocalDateTimeApi()  {        // the current date      LocalDate date = LocalDate.now();      System.out.println("the current date is "+                          date);          // the current time      LocalTime time = LocalTime.now();      System.out.println("the current time is "+                          time);          // will give us the current time and date      LocalDateTime current = LocalDateTime.now();      System.out.println("current date and time : "+                          current);          // to print in a particular format      DateTimeFormatter format =        DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");        String formatedDateTime = current.format(format);        System.out.println("in formatted manner "+                          formatedDateTime);          // printing months days and seconds      Month month = current.getMonth();      int day = current.getDayOfMonth();      int seconds = current.getSecond();      System.out.println("Month : "+month+" day : "+                          day+" seconds : "+seconds);        // printing some specified date      LocalDate date2 = LocalDate.of(1950,1,26);      System.out.println("the republic day :"+date2);        // printing date with current time.      LocalDateTime specificDate =          current.withDayOfMonth(24).withYear(2016);        System.out.println("specific date with "+                         "current time : "+specificDate);  }        // Driver code      public static void main(String[] args)      {          LocalDateTimeApi();      }  } |

Output

the current date is 2021-09-23

the current time is 20:52:39.954238

current date and time : 2021-09-23T20:52:39.956909

in formatted manner 23-09-2021 20:52:39

Month : SEPTEMBER day : 23 seconds : 39

the republic day :1950-01-26

specific date with current time : 2016-09-24T20:52:39.956909

Zoned date-time API : Use it when time zones are to be considered

Java

|  |
| --- |
| // Java code for Zoned date-time API  import java.time.LocalDateTime;  import java.time.ZoneId;  import java.time.ZonedDateTime;  import java.time.format.DateTimeFormatter;    public class Zone {    // Function to get Zoned Date and Time  public static void ZonedTimeAndDate()  {      LocalDateTime date = LocalDateTime.now();      DateTimeFormatter format1 =        DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");        String formattedCurrentDate = date.format(format1);        System.out.println("formatted current Date and"+                        " Time : "+formattedCurrentDate);        // to get the current zone      ZonedDateTime currentZone = ZonedDateTime.now();      System.out.println("the current zone is "+                          currentZone.getZone());        // getting time zone of specific place      // we use withZoneSameInstant(): it is      // used to return a copy of this date-time      // with a different time-zone,      // retaining the instant.      ZoneId tokyo = ZoneId.of("Asia/Tokyo");        ZonedDateTime tokyoZone =              currentZone.withZoneSameInstant(tokyo);        System.out.println("tokyo time zone is " +                          tokyoZone);        DateTimeFormatter format =          DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");        String formatedDateTime = tokyoZone.format(format);        System.out.println("formatted tokyo time zone "+                          formatedDateTime);    }        // Driver code      public static void main(String[] args)      {            ZonedTimeAndDate();        }  } |

Output:

formatted current Date and Time : 09-04-2018 06:21:13

the current zone is Etc/UTC

tokyo time zone is 2018-04-09T15:21:13.220+09:00[Asia/Tokyo]

formatted tokyo time zone 09-04-2018 15:21:13

Period and Duration classes :   
Period : It deals with date based amount of time.   
Duration : It deals with time based amount of time.

Java

|  |
| --- |
| // Java code for period and duration  import java.time.LocalDate;  import java.time.LocalTime;  import java.time.Month;  import java.time.Duration;  import java.time.Period;    public class Geekforgeeks {        public static void checkingPeriod()      {          LocalDate date1 = LocalDate.now();            LocalDate date2 =              LocalDate.of(2014, Month.DECEMBER, 12);            Period gap = Period.between(date2, date1);          System.out.println("gap between dates "+                             "is a period of "+gap);  }        // Function to check duration      public static void checkingDuration()      {            LocalTime time1 = LocalTime.now();          System.out.println("the current time is " +                              time1);            Duration fiveHours = Duration.ofHours(5);            // adding five hours to the current          // time and storing it in time2          LocalTime time2 = time1.plus(fiveHours);            System.out.println("after adding five hours " +                             "of duration " + time2);            Duration gap = Duration.between(time2, time1);          System.out.println("duration gap between time1" +                             " & time2 is " + gap);  }        // Driver code      public static void main(String[] args)      {          checkingPeriod();          checkingDuration();      }  } |

Output

gap between dates is a period of P6Y6M25D

the current time is 18:34:24.813548

after adding five hours of duration 23:34:24.813548

duration gap between time1 & time2 is PT-5H

ChronoUnits Enum : java.time.temporal.ChronoUnit enum is added in Java 8 to replace integer values used in old API to represent day, month etc.

Java

|  |
| --- |
| // Java code for ChronoUnits Enum  import java.time.LocalDate;  import java.time.temporal.ChronoUnit;    public class Geeksforgeeks {        // Function to check ChronoUnit      public static void checkingChronoEnum()      {          LocalDate date = LocalDate.now();          System.out.println("current date is :" +                              date);            // adding 2 years to the current date          LocalDate year =               date.plus(2, ChronoUnit.YEARS);            System.out.println("next to next year is " +                              year);            // adding 1 month to the current data          LocalDate nextMonth =                    date.plus(1, ChronoUnit.MONTHS);            System.out.println("the next month is " +                              nextMonth);            // adding 1 week to the current date          LocalDate nextWeek =                    date.plus(1, ChronoUnit.WEEKS);            System.out.println("next week is " + nextWeek);            // adding 2 decades to the current date          LocalDate Decade =                    date.plus(2, ChronoUnit.DECADES);            System.out.println("20 years after today " +                              Decade);      }        // Driver code      public static void main(String[] args) {            checkingChronoEnum();        }  } |

Output:

current date is :2018-04-09

next to next year is 2020-04-09

the next month is 2018-05-09

next week is 2018-04-16

20 years after today 2038-04-09

TemporalAdjuster : It is used to perform various date related operations.

Java

|  |
| --- |
| // Java code Temporal Adjuster  import java.time.LocalDate;  import java.time.temporal.TemporalAdjusters;  import java.time.DayOfWeek;    public class Geek  {        // Function to check date and time      // according to our requirement      public static void checkingAdjusters()      {            LocalDate date = LocalDate.now();          System.out.println("the current date is "+                              date);            // to get the first day of next month          LocalDate dayOfNextMonth =                date.with(TemporalAdjusters.                          firstDayOfNextMonth());            System.out.println("firstDayOfNextMonth : " +                              dayOfNextMonth );            // get the next saturday          LocalDate nextSaturday =                  date.with(TemporalAdjusters.                            next(DayOfWeek.SATURDAY));            System.out.println("next saturday from now is "+                              nextSaturday);            // first day of current month          LocalDate firstDay =                    date.with(TemporalAdjusters.                    firstDayOfMonth());            System.out.println("firstDayOfMonth : " +                              firstDay);            // last day of current month          LocalDate lastDay =                    date.with(TemporalAdjusters.                              lastDayOfMonth());            System.out.println("lastDayOfMonth : " +                              lastDay);  }        // Driver code      public static void main(String[] args)      {            checkingAdjusters();      }  } |

Output

the current date is 2021-07-09

firstDayOfNextMonth : 2021-08-01

next saturday from now is 2021-07-10

firstDayOfMonth : 2021-07-01

lastDayOfMonth : 2021-07-31