Date e orari

Le date sono degli oggetti molto complesse da gestire: assumono forme diverse a seconda del luogo geografico in cui ci troviamo. La manipolazione delle date e delle ore è una delle attività ricorrenti di un programmatore.

Se lavoriamo con Java 7, a classe principale per gestire date e orari è Calendar (che ha sostituito la classe Date, dichiarata **deprecata__).

Altre classi utili sono **GregorianCalendar**, **SimpleTimeZone** e **SimpleTimeZone**. Inoltre sono disponibili le più moderne LocalDate e LocalDateTime. Oltre a queste classi, è molto probabile che serva utilizzarne altre quali: **DateFormat** e **SimpleDateFormat**, che permettono la trasformazione da stringa a data e viceversa.

Novità in Java 8: LocalDate, LocalTime, LocalDateTime

```
	LocalDate oggi = LocalDate.now(); // Data di oggi
	System.out.println("oggi: " + oggi);

	//settare una data precisa (mese 1-based)
	LocalDate mauroBirthday = LocalDate.of(1969, 7, 28);
	//Per il mese possiamo usare le costanti di Month
	mauroBirthday = LocalDate.of(1969, Month.JULY, 28);

	System.out.println("mauroBirthday: " + mauroBirthday);

	LocalDate inizioCorsi = LocalDate.of(2017, Month.OCTOBER, 12);
	LocalDate natale = LocalDate.of(2017, Month.DECEMBER, 25);

	System.out.println("Fino a  natale: " + inizioCorsi.until(natale));
	System.out.println("Fino a  natale: " + inizioCorsi.until(natale,
ChronoUnit.DAYS));

	LocalDate festaLiberazione = LocalDate.of(2017, Month.APRIL, 25);
	LocalDate natale = LocalDate.of(2017, Month.DECEMBER, 25);

	System.out
		.println("Fino a  natale: " + festaLiberazione.until(natale));
	System.out.println("Fino a  natale: "
		+ festaLiberazione.until(natale, ChronoUnit.DAYS));

	System.out.println(oggi.plusMonths(1));
	System.out.println(oggi.minusMonths(1));

	DayOfWeek inizioMillenio = LocalDate.of(2000, 1, 1).getDayOfWeek();
	System.out.println("inizioMillenio: " + inizioMillenio);
	System.out.println(inizioMillenio.getValue());
	System.out.println(DayOfWeek.SATURDAY.plus(3));
```

```
        LocalDateTime ldt = LocalDateTime.now();
        System.out.println(ldt);

        LocalDate ld = LocalDate.of(2009, 1, 28);
        System.out.println(ld);

        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("M/d/yyyy");
        System.out.println(dtf.format(ld));
```

Convertire LocalDate a java.sql.Date

```
import java.sql.Date;
//...
LocalDate locald = LocalDate.of(1969, 07, 28);
Date date = Date.valueOf(locald);
r.setDateOfBirth(date);
```

---

l'operazione contraria è

```
Date date = r.getDate();
LocalDate localD = date.toLocalDate();
```

r è il record e .getDate() il metodo per farsi ritornare la data. Se avessi un campo dataNascita ilmetodo dovrebbe chiamarsi getDateNascita().

Usa le classi del package java.time invece di java.util.Date e java.sql.Date con JDBC 4.2 o superiore.

Esempio con PreparedStatement

```
myPreparedStatement.setObject(
    … ,                                         // qui passa il numero
ordinale dell'argomento.
    myJavaUtilDate.toInstant()                  // Converti da
`java.util.Date` nel più moderno `java.time.Instant` (UTC).
        .atZone( ZoneId.of( "Europe/Rome" ) )   // Sette un time zone
particolare, per determinare la data. Instanziando un `ZonedDateTime`.
        .toLocalDate()                          // Estrai la data di tipo
`java.time.LocalDate` dall'oggetto.
)
```

---

esempi

```
LocalDate todayLocalDate = LocalDate.now( ZoneId.of( "Europe/Paris" ) );
// Usare "continent/region" come region name; non usare quelli codificati
da 3 lettere.

LocalDate localDate = ResultSet.getObject( 1 , LocalDate.class );

//la questione è irrelevante in JDBC 4.2 o successive.
```

Converti a java.sql.Date

```
java.sql.Date sqlDate = java.sql.Date.valueOf( todayLocalDate );

//viceversa

LocalDate localDate = sqlDate.toLocalDate();
```

Nuovi metodi di java.util.Date

- java.util.Date.from(Instant)
- java.util.Date::toInstant.

```
Instant instant = myUtilDate.toInstant();

ZoneId zoneId = ZoneId.of ( "America/Montreal" );
ZonedDateTime zdt = ZonedDateTime.ofInstant ( instant , zoneId );
LocalDate localDate = zdt.toLocalDate();
```

```
public class MainClass {

  public static void main(String[] args) {
    java.util.Date utilDate = new java.util.Date();
    java.sql.Date sqlDate = new java.sql.Date(utilDate.getTime());
    System.out.println("utilDate:" + utilDate);
    System.out.println("sqlDate:" + sqlDate);

  }

}

//http://www.java2s.com/Tutorial/Java/0040__Data-
Type/ConvertfromajavautilDateObjecttoajavasqlDateObject.htm
```

GreorianCalendar

La classe **GregorianCalendar** è molto semplice da utilizzare.

Sono disponibili diversi costruttori. Il costruttore senza parametri inizializza l'oggetto con la data e l'ora attuale. Con il metodo get(), ereditato da Calendar, è possibile ricevere tutte le informazioni disponibili per l'oggetto di tipo data.

Primo esempio

stampiamo semplicemente la data odierna con l'orario attuale.

```
GregorianCalendar calendario = new GregorianCalendar();
int anno = calendario.get(Calendar.YEAR);
int mese = calendario.get(Calendar.MONTH) + 1;
int giorno = calendario.get(Calendar.DATE);
int ore = calendario.get(Calendar.HOUR);
int min = calendario.get(Calendar.MINUTE);
int sec = calendario.get(Calendar.SECOND);

System.out.println(giorno + "/" + mese + "/" + anno);
System.out.println(ore + ":" + minuti + ":" + secondi);
```

Formattare la data: SimpleDateFormat

la classe SimpleDateFormat che permette di trattare le date nel formato più adatto alla nostra esigenza.

Secondo esempio

come stampare la data odierna usando **SimpleDateFormat** per formattare l'output.

```
GregorianCalendar calendario = new GregorianCalendar();
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yy - HH:mm:ss");
System.out.println(sdf.format(calendario.getTime()));
```

```
- Il costruttore della classe SimpleDateFormat prende in ingresso una
stringa che rappresenta il formato della data che vogliamo stampare.
- Il metodo getTime() della classe GregorianCalendar restituisce un'istanza
di Date.
- Il metodo format() della classe SimpleDateFormat, che restituisce in
ingresso una Date, restituisce una stringa che corrisponde al formato che
```

```
abbiamo impostato.
- E' possibile sfruttare la classe SimpleDateFormat anche per ottenere
un'istanza della classe Calendar.
```

```
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yy - HH:mm:ss");
String miaData = "15/04/1988";
GregorianCalendar calendario = new GregorianCalendar();
try {
    calendario.setTime(sdf.parse(miaData));
} catch(ParseException exc) {
    exc.printStackTrace();
}
```

```
Il metodo __parse()__ della classe SimpleDateFormat riceve in ingresso una
stringa e restituisce un oggetto Date.
Il metodo __setTime()__ della classe GregorianCalendar ci permette di
impostare la data.
```

Bisogna utilizzare un blocco try-catch perché potrebbe essere sollevata una **ParseException**, nel caso in cui una stringa passata al metodo parse(), non rappresenti una data convertibile.

Terzo esempio

come convertire una data dal formato americano in quello italiano utilizzando le tecniche analizzate in precedenza.

```
SimpleDateFormat formatIT = new SimpleDateFormat("dd/MM/yyyy");
SimpleDateFormat formatUS = new SimpleDateFormat("yyyy/MM/dd");

Date dataIT;
try {
    dataIT = formatUS.parse("2017/12/25");
    String dataUS = formatIT.format(dataIT);
    System.out.println(dataUS);
} catch(ParseException exc) {
    exc.printStackTrace();
}
```

Volendo confrontare due date possiamo utilizzare i metodi - **after()** - **before()** - **equals()** presenti nella classe Date.

```
GregorianCalendar c1 =
new gregorianCalendar(2013, GregorianCalendar.FEBRUARY, 05);
```

```
GregorianCalendar c2 =
new gregorianCalendar(2013, GregorianCalendar.FEBRUARY, 05);

Date data1 = c1.getTime();
Date data2 = c2.getTime();
```

Il metodo:

```
data1.after(data2)  restituirà false
data1.equals(data2) restituirà false
data1.before(data2) restituirà true
```

Java Legacy

I membri resi disponibili dalla classe Date.

Costruttori pubblici:

..	..
Date()	Costruisce un oggetto Date che incapsula la data e l'ora correnti.
Date(long t)	Costruisce un oggetto Date che incapsula la data e l'ora espressi dall'argomento t. L'argomento è un long che riporta i millisecondi trascorsi dal 1è Gennaio 1970 sino alla data rappresentata.

Metodi pubblici:

..	..
boolean after(Date d)	Restituisce true se la data di invocazione è successiva alla data d.
boolean before(Date d)	Restituisce true se la data di invocazione è precedente alla data d.
Object clone()	Clona l'oggetto. Date implementa l'interfaccia Cloneable.
int compareTo(Date d)	Compara la data di invocazione con d. Restituisce 0 se le due date sono uguali, un valore negativo se la data di invocazione precede la data d o un valore positivo se la data di invocazione è successiva alla data d.
int compareTo(Object)	Se l'argomento fornito è una istanza di Date, agisce come compareTo(Date d). In caso contrario, propaga una ClassCastException.
boolean equals(Object o)	Restituisce true se l'argomento è una data equivalente a quella di invocazione.

..	..
long getTime()	Restituisce la data dell'oggetto sotto forma di valore long, che esprime i millisecondi trascorsi dal 1 Gennaio 1970 sino alla data rappresentata.
int hashCode()	Calcola un codice hash per l'oggetto.
void setTime(long t)	Imposta la data rappresentata con un argomento di tipo long, che esprime i millisecondi trascorsi dal 1 Gennaio 1970 sino alla data rappresentata.
String toString()	Fornisce una rappresentazione in stringa della data.

Classe GregorianCalendar.

Costruttori pubblici:

GregorianCalendar Costruisce un GregorianCalendar che rappresenta la data e l'ora correnti.

..	..
GregorianCalendar(int year, int month, int date)	Costruisce un GregorianCalendar che rappresenta la data espressa mediante gli argomenti forniti. GregorianCalendar(int year, int month, int date,int hour, int minute)

Metodi pubblici:

..	..
boolean after(Object o)	Restituisce true se la data rappresentata È successiva alla data espressa dall'oggetto o, che deve essere istanza di Calendar.
boolean before(Object o)	Restituisce true se la data rappresentata È precedente alla data espressa dall'oggetto o, che deve essere istanza di Calendar.
Object clone()	Clona l'oggetto. GregorianCalendar implementa l'interfaccia Cloneable.
booleans equals(Object o)	Restituisce true se l'argomento rappresenta una data equivalente a quella rappresentata dall'oggetto di invocazione.
int get(int field)	Recupera il valore di uno dei campi della data rappresentata. L'argomento specifica il campo di interesse.
Date getTime()	Restituisce un oggetto Date che rappresenta la data incapsulata dall'oggetto di invocazione.
long getTimeInMillis()	Restituisce la data rappresentata dall'oggetto sotto forma di valore long, che esprime i millisecondi trascorsi dal 1 Gennaio 1970 sino alla data rappresentata.
int hashCode()	Calcola un codice hash per l'oggetto.
void set(int field, int value)	Imposta su value il valore del campo rappresentato dall'intero field.

..	..
void setTime(Date d)	Imposta la data rappresentata prelevando il suo valore all'argomento d.
void SetTimeInMillis(long l)	Imposta la data rappresentata mediante un argomento di tipo long, che esprime i millisecondi trascorsi dal 1 Gennaio 1970 sino alla data rappresentata.
String toString()	Fornisce una rappresentazione in stringa della data. rappresentata.

Costanti statiche:

..	..
AM_PM	Il campo che informa se l'ora espressa è prima o dopo mezzogiorno.
DAY_OF_MONTH	Il campo che riporta il giorno del mese.
DAY_OF_WEEK	Il campo che riporta il giorno della settimana.
DAY_OF_YEAR	Il campo che riporta il giorno dell'anno.
HOUR	Il campo che riporta l'ora del mattino o del pomeriggio, a seconda del contenuto del campo AM_PM
HOUR_OF_DAY	Il campo che riporta l'ora del giorno, in un intervallo tra 0 e 23.
MILLISECOND	Il campo che riporta i millisecondi.
MINUTE	Il campo che riporta i minuti.
MONTH	Il campo che riporta il mese.
SECOND	Il campo che riporta i secondi.
WEEK_OF_MONTH	Il campo che riporta la settimana del mese.
WEEK_OF_YEAR	Il campo che riporta la settimana dell'anno.
YEAR	Il campo che riporta l'anno.