Date và Time trong Java

Java cung cấp lớp **Date** có sẵn trong **java.util** package, lớp này tóm lược ngày tháng và thời gian hiện tại.

Để hiểu sâu hơn các khái niệm được trình bày trong chương này, mời bạn tham khảo loạt bài: **Ví dụ về Date & Time trong Java**.

Lớp Date hỗ trợ hai constructor. Constructor đầu tiên khởi tạo đối tượng với ngày và thời gian hiện tại.

```
Date( )
```

Constructor sau chấp nhận một tham số bằng số mili giây đã trôi qua từ nửa đêm ngày 1/1/1970.

```
Date(long millisec)
```

Một khi bạn có một đối tượng Date có sẵn, bạn có thể gọi bất kỳ phương thức hỗ trợ nào để thao tác với ngày tháng này:

STT	Phương thức và Miêu tả
1	**boolean after(Date date)** Trả về true nếu gọi đối tượng Date chứa một ngày mà chậm hơn ngày đã xác định, nếu không là false.
2	**boolean before(Date date)** Trả về true nếu gọi đối tượng Date chứa một ngày mà sớm hơn ngày đã xác định, nếu không là false.
3	**Object clone( )** Sao chép đối tượng Date đang gọi
4	**int compareTo(Date date)** So sánh giá trị đối tượng đang gọi với giá trị đó của date. Trả về 0 nếu các giá trị này

	là cân bằng. Trả về một giá trị âm nếu đối tượng đang gọi là sớm hơn date. Trả về một giá trị dương nếu đối tượng đang gọi chậm hơn date.
5	**int compareTo(Object obj)** Tiến hành tương tự như compareTo(Date) nếu đối tượng là của lớp Date. Nếu không thì, nó cho một ClassCastException.
6	**boolean equals(Object date)** Trả về true nếu đối tượng Date đang gọi chứa thời gian và ngày tháng giống như date đã cho, nếu không là false.
7	**long getTime( )** Trả về số mili giây đã trôi qua từ 1/1/1970
8	**int hashCode( )** Trả về một mã hóa băm (hash code) cho đối tượng đang gọi
9	**void setTime(long time)** Thiết lập ngày tháng và thời gian như time đã cho, mà biểu diễn một time đã trôi qua (giá trị mili giây) từ nửa đêm 1/1/1970
10	**String toString( )** Biến đổi đối tượng Date đang gọi thành một chuỗi và trả về kết quả

Nhận Date và Time hiện tại trong Java

Thực sự dễ dàng để nhận date và time hiện tại trong Java. Bạn có thể sử dụng một đối tượng Date đơn giản với phương thức *toString()* để in date và time hiện tại như sau:

```
import java.util.Date;


public class DateDemo {
```

```
   public static void main(String args[]) {
       // Instantiate a Date object
       Date date = new Date();

       // display time and date using toString()
       System.out.println(date.toString());
   }
}
```

Nó sẽ cho kết quả sau:

```
Mon May 04 09:51:52 CDT 2009
```

So sánh Date trong Java

Có 3 cách để so sánh hai date trong Java:

- Bạn có thể sử dụng getTime() để nhận số mili giây đã trôi qua từ nửa đêm 1/1/1970 cho cả hai đối tượng và sau đó so sánh hai giá trị này.
- Bạn có thể sử dụng các phương thức before(), after() và equals(). Bởi vì tháng thứ 12 ở trước tháng thứ 18, ví dụ, new Date(99, 2, 12).before(new Date(99, 2, 18)) trả về true.
- Bạn có thể sử dụng phương thức compareTo(), mà được định nghĩa bởi Comparable interface và được thi hành bởi Date.

Định dạng Date bởi sử dụng SimpleDateFormat trong Java

SimpleDateFormat là một lớp cố định (cụ thể) để định dạng và parse các date theo một phương thức nhạy cảm với locale. SimpleDateFormat cho phép bạn bắt đầu bởi việc họn bất kỳ pattern đã được định nghĩa bởi người dùng cho định dạng date-time. Ví dụ:

```
import java.util.*;
import java.text.*;

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

```
      Date dNow = new Date( );
      SimpleDateFormat ft =
      new SimpleDateFormat ("E yyyy.MM.dd 'at' hh:mm:ss a zzz");

      System.out.println("Current Date: " + ft.format(dNow));
   }
}
```

Nó sẽ cho kết quả:

```
Current Date: Sun 2004.07.18 at 04:14:09 PM PDT
```

Mã hóa định dạng SimpleDateFormat trong Java

Để xác định định dạng thời gian, sử dụng một chuỗi time mẫu. Trong pattern này, tất cả chữ cái ASCII được dự trữ (dành riêng) như là các ký tự pattern, mà được định nghĩa như sau:

Ký tự	Miêu tả	Ví dụ
G	Tên mệnh danh của thời đại	AD
y	Năm trong dạng 4 ký số	2001
M	Tháng trong năm	July or 07
d	Ngày trong tháng	10
h	Giờ trong dạng A.M./P.M. (1~12)	12
H	Giờ trong ngày (0~23)	22
m	Phút trong giờ	30

s	Giây trong phút	55
S	Số mili giây	234
E	Ngày trong tuần	Tuesday
D	Ngày trong năm	360
F	Ngày của tuần trong tháng	2 (Wed thứ hai trong July)
w	Tuần trong năm	40
W	Tuần trong tháng	1
a	A.M./P.M.	PM
k	Giờ trong ngày (1~24)	24
K	Giờ dạng A.M./P.M. (0~11)	10
z	Time zone	Eastern Standard Time
'	Dãy thoát cho văn bản	Dấu giới hạn
"	Trích dẫn đơn	`

Định dạng Date sử dụng printf trong Java

Định dạng date và time có thể được thực hiện một cách đơn giản bởi sử dụng phương thức**printf** trong Java. Bạn sử dụng một định dạng hai chữ cái, bắt đầu với **t** và kết thúc với một trong các ký tự trong bảng dưới. Ví dụ:

```
import java.util.Date;

public class DateDemo {

  public static void main(String args[]) {
     // Instantiate a Date object
     Date date = new Date();

     // display time and date using toString()
     String str = String.format("Current Date/Time : %tc", date );

     System.out.printf(str);
  }
}
```

Nó sẽ cho kết quả:

```
Current Date/Time : Sat Dec 15 16:37:57 MST 2012
```

Nó sẽ là khá ngờ nghệch nếu bạn phải cung cấp cho date nhiều time để định dạng mỗi phần. Với lý do này, một chuỗi định dạng có thể chỉ dẫn chỉ mục của tham số để được định dạng.

Chỉ mục phải ngay lập tức theo sau bởi dấu % và nó phải được kết thúc bởi một dấu $. Ví dụ:

```
import java.util.Date;

public class DateDemo {

   public static void main(String args[]) {
       // Instantiate a Date object
       Date date = new Date();

       // display time and date using toString()
       System.out.printf("%1$s %2$tB %2$td, %2$tY",
                         "Due date:", date);
```

```
   }
}
```

Nó sẽ cho kết quả:

```
Due date: February 09, 2004
```

Bạn cũng có thể sử dụng ký hiệu <. Nó chỉ dẫn tham số tương tự như trong định dạng trước. Ví dụ:

```
import java.util.Date;


public class DateDemo {


   public static void main(String args[]) {
       // Instantiate a Date object
       Date date = new Date();


       // display formatted date
       System.out.printf("%s %tB %<te, %<tY",
                         "Due date:", date);
   }
}
```

Nó sẽ cho kết quả:

```
Due date: February 09, 2004
```

Các ký tự biến đổi Date và Time trong Java

Ký tự	Miêu tả	Ví dụ
c	Ngày tháng đầy đủ	Mon May 04 09:51:52 CDT 2009
F	Định dạng ngày ISO 8601	2004-02-09

D	Định dạng ngày theo U.S. (month/day/year)	02/09/2004
T	Thời gian dạng 24 giờ	18:05:19
r	Thời gian dạng 12 giờ	06:05:19 pm
R	Thời gian dạng 24 giờ, không có giây	18:05
Y	Số năm 4 ký số (bắt đầu từ 0)	2004
y	2 số cuối của năm (bắt đầu từ 0)	04
C	2 số đầu của năm (bắt đầu từ 0)	20
B	Tên tháng đầy đủ	February
b	Tên tháng viết tắt	Feb
m	Tháng dạng 2 ký số (bắt đầu từ 0)	02
d	Ngày dạng 2 ký số (bắt đầu từ 0)	03
e	Ngày dạng hai ký số (không bắt đầu từ 0)	9
A	Tên ngày trong tuần đầy đủ	Monday
a	Tên ngày trong tuần viết tắt	Mon
j	Ngày trong năm dạng 3 ký số (bắt đầu từ 0)	069

H	Giờ dạng hai ký số (bắt đầu từ 0), giữa 0 và 23	18
k	Giờ dạng hai ký số (không bắt đầu từ 0), giữa 0 và 23	18
I	Giờ dạng hai ký số (bắt đầu từ 0), giữa 0 và 12	06
l	Giờ dạng hai ký số (không bắt đầu từ 0), giữa 0 và 12	6
M	Phút dạng hai ký số (bắt đầu từ 0)	05
S	Giây dạng hai ký số (bắt đầu từ 0)	19
L	Mili giây dạng ba ký số (bắt đầu từ 0)	047
N	Nano giây dạng 9 ký số (bắt đầu từ 0)	047000000
P	Sáng hoặc chiều dạng chữ hoa	PM
p	Sáng hoặc chiều dạng chữ thường	pm
z	Offset dạng số RFC 822 từ GMT	-0800
Z	Time zone	PST
s	Số giây từ 1/1/1970 00:00:00 GMT	1078884319
Q	Số mili giây từ 1/1/1970 00:00:00 GMT	1078884319047

Có một số lớp hữu ích khác liên quan tới Date và time. Để biết thêm chi tiết, bạn tham khảo văn kiện chuẩn của Java (Java Standard Documentation).

Parse các String vào trong các Date trong Java

Lớp SimpleDateFormat có một số phương thức bổ sung, đáng kể nhất là **parse()**, mà parse một chuỗi theo định dạng được lưu giữ trong đối tượng SimpleDateFormat đã cho. Ví dụ:

```
import java.util.*;
import java.text.*;

public class DateDemo {

   public static void main(String args[]) {
      SimpleDateFormat ft = new SimpleDateFormat ("yyyy-MM-dd");

      String input = args.length == 0 ? "1818-11-11" : args[0];

      System.out.print(input + " Parses as ");

      Date t;

      try {
          t = ft.parse(input);
          System.out.println(t);
      } catch (ParseException e) {
          System.out.println("Unparseable using " + ft);
      }
   }
}
```

Chạy mẫu chương trình trên sẽ cho kết quả sau:

```
$ java DateDemo
1818-11-11 Parses as Wed Nov 11 00:00:00 GMT 1818
$ java DateDemo 2007-12-01
2007-12-01 Parses as Sat Dec 01 00:00:00 GMT 2007
```

Đình chỉ trong chốc lát (sleep for a while) trong Java

Bạn có thể ngừng bất kỳ chu kỳ thời gian nào từ một mili giây tới một vòng đời lifetime của máy tính. Ví dụ, chương trình sau sẽ đình chỉ trong 10 giây:

```
import java.util.*;


public class SleepDemo {
   public static void main(String args[]) {
      try {
         System.out.println(new Date( ) + "\n");
         Thread.sleep(5*60*10);
         System.out.println(new Date( ) + "\n");
      } catch (Exception e) {
          System.out.println("Got an exception!");
      }
   }
}
```

Nó sẽ cho kết quả:

```
Sun May 03 18:04:41 GMT 2009


Sun May 03 18:04:51 GMT 2009
```

Đo lường thời gian đã trôi qua trong Java

Đôi khi, bạn có thể cần đo lường thời gian tại một thời điểm với giá trị mili giây. Chúng ta viết lại ví dụ trên như sau:

```
import java.util.*;


public class DiffDemo {


   public static void main(String args[]) {
      try {
         long start = System.currentTimeMillis( );
```

```
         System.out.println(new Date( ) + "\n");
         Thread.sleep(5*60*10);
         System.out.println(new Date( ) + "\n");
         long end = System.currentTimeMillis( );
         long diff = end - start;
         System.out.println("Difference is : " + diff);
      } catch (Exception e) {
         System.out.println("Got an exception!");
      }
   }
}
```

Nó sẽ cho kết quả:

```
Sun May 03 18:16:51 GMT 2009

Sun May 03 18:16:57 GMT 2009

Difference is : 5993
```

Lớp GregorianCalendar trong Java

GregorianCalendar là một sự bổ sung cụ thể của một lớp Calendar mà thực hiện lịch Gregorian chuẩn. Chúng tối không đề cập lớp Calendar trong loạt bài này, bạn có thể tìm hiểu nó trong văn kiện chuẩn của Java.

Phương thức *getInstance()* của Calendar trả về một GregorianCalendar được khởi tạo với date và time hiện tại theo localde và time zone mặc định. GregorianCalendar định nghĩa hai trường: AD và BC. Chúng biểu diễn hai thời đại được định nghĩa bởi GregorianCalendar.

Cũng có một số constructor cho các đối tượng GregorianCalendar:

STT	Constructor và miêu tả
1	**GregorianCalendar()**

	Xây dựng một GregorianCalendar mặc định bởi sử dụng time hiện tại trong time zone mặc định với locale mặc định.
2	**GregorianCalendar(int year, int month, int date)** Xây dựng một GregorianCalendar với thiết lập date đã cho trong time zone mặc định với locale mặc định.
3	**GregorianCalendar(int year, int month, int date, int hour, int minute)** Xây dựng một GregorianCalendar với thiết lập date và time đã cung cấp cho time zone mặc định với locale mặc định.
4	**GregorianCalendar(int year, int month, int date, int hour, int minute, int second)** Xây dựng một GregorianCalendar với thiết lập date và time đã cung cấp cho time zone mặc định với locale mặc định.
5	**GregorianCalendar(Locale aLocale)** Xây dựng một GregorianCalendar trên cơ sở time hiện tại trong time zone mặc định với locale đã cho
6	**GregorianCalendar(TimeZone zone)** Xây dựng một GregorianCalendar trên cơ sở time hiện tại trong time zone đã cung cấp với locale mặc định
7	**GregorianCalendar(TimeZone zone, Locale aLocale)** Xây dựng một GregorianCalendar trên cơ sở time hiện tại trong time zone đã cung cấp với locale đã cho

Dưới đây liệt kê một số phương thức hỗ trợ hữu ích được cung cấp bởi lớp GregorianCalendar trong Java:

STT	Phương thức và miêu tả
1	**void add(int field, int amount)** Thêm số time đã xác định tới trường time đã cung cấp, dựa trên các qui tắc của calendar đó.
2	**protected void computeFields()** Biến đổi UTC dạng mili giây thành các giá trị trường time.
3	**protected void computeTime()** Override các giá trị biến đổi trường time thành UTC dạng mili giây.
4	**boolean equals(Object obj)** So sánh GregorianCalendar với một tham chiếu đối tượng
5	**int get(int field)** Nhận giá trị cho một trường time đã cung cấp
6	**int getActualMaximum(int field)** Trả về giá trị lớn nhất mà trường này có thể có, đã cung cấp date hiện tại
7	**int getActualMinimum(int field)** Trả về giá trị nhỏ nhất mà trường này có thể có, đã cung cấp date hiện tại
8	**int getGreatestMinimum(int field)** Trả về giá trị tối thiểu cao nhất cho trường đã cung cấp nếu nó biến đổi
9	**Date getGregorianChange()**

	Nhận date thay đổi theo Gregorian Calendar
10	**int getLeastMaximum(int field)** Trả về giá trị tối đa thấp nhất cho trường đã cung cấp nếu nó biến đổi
11	**int getMaximum(int field)** Trả về giá trị tối đa cho trường đã cung cấp
12	**Date getTime()** Nhận time hiện tại của Calendar này
13	**long getTimeInMillis()** Nhận time hiện tại của Calendar này dạng long
14	**TimeZone getTimeZone()** Nhận time zone.
15	**int getMinimum(int field)** Trả về giá trị tối thiểu cho trường đã cung cấp
16	**int hashCode()** Override mã hóa băm.
17	**boolean isLeapYear(int year)** Xác định nếu năm đã cho là một leap year
18	**void roll(int field, boolean up)** Cộng hoặc trừ một đơn vị đơn của time trên trường time đã cho mà không thay đổi các trường lớn hơn

19	**void set(int field, int value)** Thiết lập trường time với giá trị đã cho
20	**void set(int year, int month, int date)** Thiết lập các giá trị cho các trường year, month, và date.
21	**void set(int year, int month, int date, int hour, int minute)** Thiết lập các giá trị cho các trường year, month, date, hour, và minute.
22	**void set(int year, int month, int date, int hour, int minute, int second)** Thiết lập các giá trị cho các trường year, month, date, hour, minute và second.
23	**void setGregorianChange(Date date)** Thiết lập ngày thay đổi theo GregorianCalendar
24	**void setTime(Date date)** Thiết lập time hiện tại của Calendar này với Date đã cho
25	**void setTimeInMillis(long millis)** Thiết lập time hiện tại của Calendar này từ giá trị long đã cho
26	**void setTimeZone(TimeZone value)** Thiết lập time zone với giá trị time zone đã cho
27	**String toString()** Trả về một biểu diễn chuỗi của calendar này

Ví dụ:

```
import java.util.*;
```

```
public class GregorianCalendarDemo {

   public static void main(String args[]) {
      String months[] = {
      "Jan", "Feb", "Mar", "Apr",
      "May", "Jun", "Jul", "Aug",
      "Sep", "Oct", "Nov", "Dec"};

      int year;
      // Create a Gregorian calendar initialized
      // with the current date and time in the
      // default locale and timezone.
      GregorianCalendar gcalendar = new GregorianCalendar();
      // Display current time and date information.
      System.out.print("Date: ");
      System.out.print(months[gcalendar.get(Calendar.MONTH)]);
      System.out.print(" " + gcalendar.get(Calendar.DATE) + " ");
      System.out.println(year = gcalendar.get(Calendar.YEAR));
      System.out.print("Time: ");
      System.out.print(gcalendar.get(Calendar.HOUR) + ":");
      System.out.print(gcalendar.get(Calendar.MINUTE) + ":");
      System.out.println(gcalendar.get(Calendar.SECOND));

      // Test if the current year is a leap year
      if(gcalendar.isLeapYear(year)) {
         System.out.println("The current year is a leap year");
      }
      else {
         System.out.println("The current year is not a leap year");
      }
   }
}
```

Nó sẽ cho kết quả sau:

```
Date: Apr 22 2009

Time: 11:25:27

The current year is not a leap year
```

Để có danh sách đầy đủ các hằng có sẵn trong lớp Calendar, bạn tham khảo văn kiện Java chuẩn.

Để hiểu sâu hơn các khái niệm được trình bày trong chương này, mời bạn tham khảo loạt bài ví dụ về Date và Time trong Java: **Ví dụ về Date & Time trong Java**.