Gérer la date et l'heure en Java

Les classes Date, DateFormat, SimpleDateFormat

La gestion des dates est indispensable dans la plupart des applications, par exemple pour mémoriser une date de naissance ou garder trace du moment où une opération a été effectuée.

1 Mémoriser la date et l'heure avec la classe Date

La classe java.util.Date représente un instant exprimé en milliseconde, dont la valeur 0 correspond au 1er janvier 1970. On utilise surtout le constructeur sans paramètre de cette classe, qui initialise la date au moment présent. De nombreux autres constructeurs sont dépréciés (voir la javadoc).

1.1 Explication préalable sur le terme « deprecated » (déprécié) de l'API Java

La bibliothèque standard Java 1.4 compte 18 classes et 281 méthodes désapprouvées pour des raisons diverses : elles sont signalées dans la documentation sur les API Java par le mot deprecated suivi d'une suggestion de remplacement. Le fait qu'une classe ou une méthode soit notée deprecated n'empêche pas de l'utiliser dans un programme mais entraîne l'affichage du message suivant lors de la compilation avertissant qu'une API désapprouvée a été utilisée : "ClasseXxx.java uses or overrides a deprecated API".
Quand un tel message s'affiche, il est conseillé de remplacer l'API désapprouvée par la solution suggérée. Toutefois, comme Sun-Oracle assure jusqu'à maintenant la compatibilité ascendante des classes Java, vous pouvez ne pas tenir compte de l'avertissement dans certains cas.

1.2 Méthodes de la classe Date

Les méthodes de cette classe appartiennent à plusieurs catégories :

Méthode	Explications
~~getYear, setYear~~ ~~getMonth, setMonth~~	**Dépréciées !!!!** ~~pour interroger et modifier l'année, le mois, le jour, les heures, les minutes et les secondes d'une instance de java.util.Date~~
getTime et setTime	pour interroger et modifier l'instant stocké par un objet de cette classe
public boolean before (Date d) public boolean after (Date d) public int compareTo (Object obj) public boolean equals (Object obj)	méthodes pour comparer des dates entre elles

La classe java.util.Date, disponible depuis leJDK 1.0, n'a pas été initialement conçue pour gérer les conversions internationales des dates. Pendant le dévelopment du JDK 1.1, Sun a décidé de « désapprouver » (to deprecate en anglais) les méthodes getYear, setYear... de cette classe (au lieu de la modifier) et a créé les classes java.text.SimpleDateFormat et java.util.CregorianCalendar pour y ajouter les fonctionnalités manquantes.

2 Formater la date avec le printf (depuis Java 5)

2.1 printf

Voici un extrait des exemples d'affichages fournis sur le site ftp :

```
System.out.printf( "%tD \n" , new Date() );
System.out.printf( "%tF \n" , new Date() );
System.out.printf( "%1$te %1$tb %1$ty \n" , new Date() );
System.out.printf( "%1$tA %1$te %1$tB %1$tY \n", new Date() );
System.out.printf( "%1$tr \n" , new Date() );
```

2.2 La méthode String.format

Avec la méthode String.format, on peut récupérer une String contenant la date au format voulu.

```
String chaineFormattee = String.format("%tF \n", new Date());
```

2.3 Utilisations de DateFormat et SimpleDateFormat

2.3.1 Première utilisation : parsing d'une chaîne pour créer une date

A partir d'une chaîne contenant l'information-date et connaissant son format (pattern), on peut construire un objet Date.
Exemple:

```
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
Date d = sdf.parse("15/10/1950");
```

Autre écriture possible:

```
Date d = new SimpleDateFormat("dd/MM/yyyy").parse("15/10/2005");
```

2.3.2 Deuxième utilisation : affichage

Pour afficher une date en suivant un format (pattern), la classe java.text.DateFormat et sa sous-classe java.text.SimpleDateFormat permettent de convertir cette date au format texte utilisé dans la langue de l'utilisateur.
Exemple :

```
SimpleDateFormat sdf = new SimpleDateFormat();
System.out.println( sdf.format(new Date()));

sdf.applyPattern("dd/MM/yyyy");
System.out.println( sdf.format(new Date()));
```

Autre écriture possible:

```
System.out.println( (DateFormat.getInstance()).format( new Date() ) ;
```

2.3.3 Détail des étapes

La procédure de conversion se déroule en deux étapes :

2.3.3.1 Etape 1 : Obtention d'une instance de format

On peut obtenir une instance de java.text.DateFormat en appelant l'une des méthodes de classe suivantes :

- getDateInstance pour utiliser le format jour, mois et année,
- getTimeInstance pour utiliser le format heures, minutes et secondes,

- getDateTimeInstance pour utiliser le format jour, mois, année, heures, minutes et secondes.

Ces méthodes prennent en paramètres les constantes FULL, LONG, MEDIUM, ou SHORT de la classe java.text.DateFormat pour spécifier le style de la conversion.

2.3.3.2 Etape 2 : Appel de la méthode format

Envoi du message « format » vers l'instance de java.text.DateFormat obtenue en lui passant en paramètre une instance de java.util.Date à formatter.

```
Locale locale = Locale.FRENCH;

// Create an instance of DateFormat.
DateFormat df = DateFormat.getDateTimeInstance(DateFormat.SHORT,
DateFormat.SHORT, locale);

if (df instanceof SimpleDateFormat) {
      SimpleDateFormat sdf = (SimpleDateFormat) df;

// Apply a pattern.
sdf.applyPattern("yyyy.MM.dd 'at' hh:mm:ss a zzz");

// Format the current time.
String dateString = sdf.format(new Date());
System.out.println(dateString + "\n");
// ex: 1997.11.27 at 05:44:10 AM GMT+01:00
```

Voir aussi les autres exemples sur le site ftp