Java Utility Classes

Sang Shin
Michèle Garoche
www.javapassion.com
“Learn with Passion!”

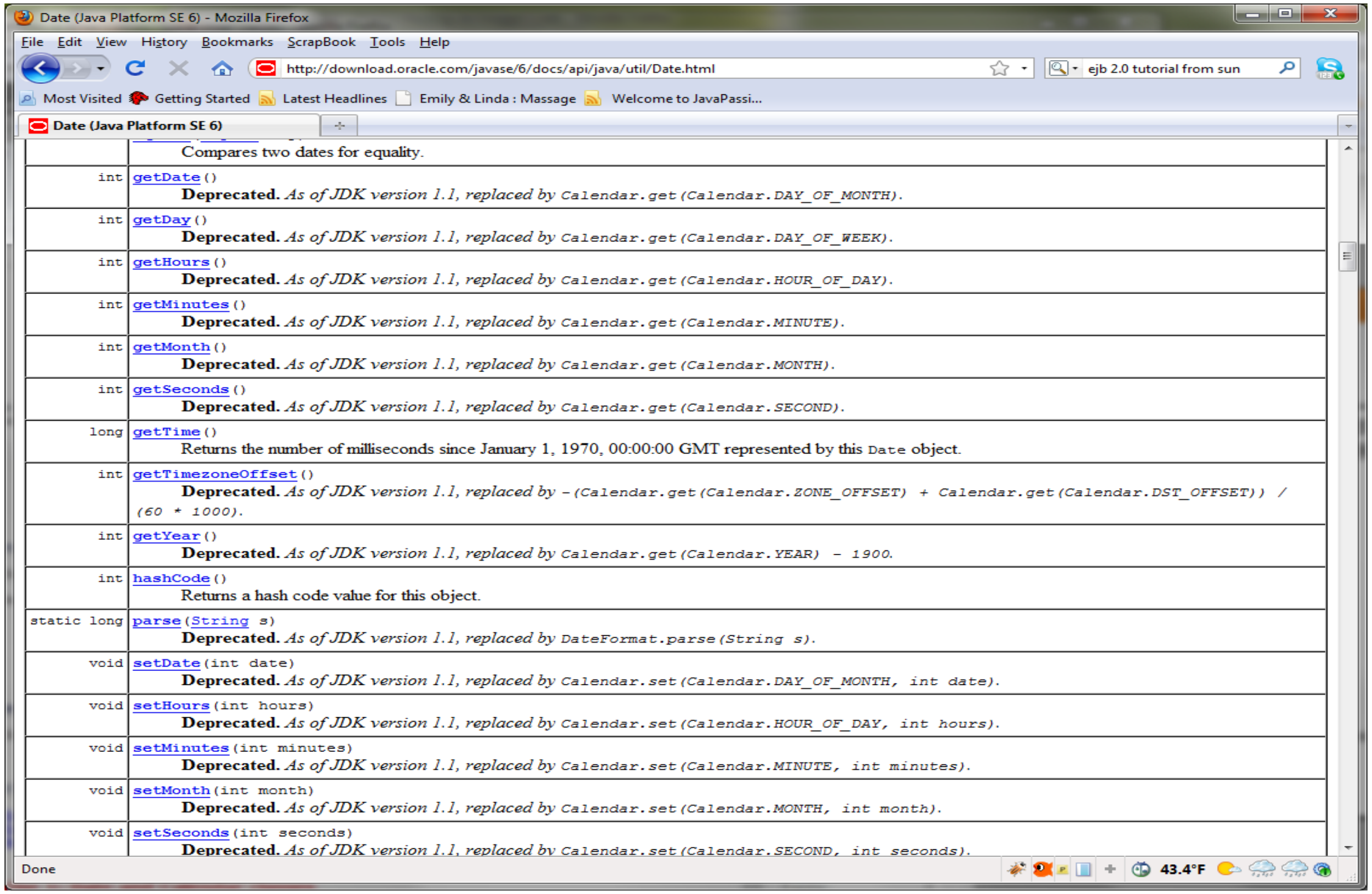
Topics

- Date class
- Calendar and GregorianCalendar classes
- SimpleDateFormat class
- Properties class

Date Class

Date Class

- Most methods of *Date* class have been deprecated from JDK 1.1 (We will talk about when you still want to use Date class later in this presentation)


Date (Java Platform SE 6) - Mozilla Firefox
File Edit View History Bookmarks ScrapBook Tools Help
http://download.oracle.com/javase/6/docs/api/java/util/Date.html
ejb 2.0 tutorial from sun
Most Visited
Getting Started
Latest Headlines
Emily & Linda : Massage
Welcome to JavaPassi...
Date (Java Platform SE 6)
Done
43.4°F

	Compares two dates for equality.
int	getDate() **Deprecated.** *As of JDK version 1.1, replaced by* `Calendar.get(Calendar.DAY_OF_MONTH)`.
int	getDay() **Deprecated.** *As of JDK version 1.1, replaced by* `Calendar.get(Calendar.DAY_OF_WEEK)`.
int	getHours() **Deprecated.** *As of JDK version 1.1, replaced by* `Calendar.get(Calendar.HOUR_OF_DAY)`.
int	getMinutes() **Deprecated.** *As of JDK version 1.1, replaced by* `Calendar.get(Calendar.MINUTE)`.
int	getMonth() **Deprecated.** *As of JDK version 1.1, replaced by* `Calendar.get(Calendar.MONTH)`.
int	getSeconds() **Deprecated.** *As of JDK version 1.1, replaced by* `Calendar.get(Calendar.SECOND)`.
long	getTime() Returns the number of milliseconds since January 1, 1970, 00:00:00 GMT represented by this `Date` object.
int	getTimezoneOffset() **Deprecated.** *As of JDK version 1.1, replaced by* `-(Calendar.get(Calendar.ZONE_OFFSET) + Calendar.get(Calendar.DST_OFFSET)) / (60 * 1000)`.
int	getYear() **Deprecated.** *As of JDK version 1.1, replaced by* `Calendar.get(Calendar.YEAR) - 1900`.
int	hashCode() Returns a hash code value for this object.
static long	parse(String s) **Deprecated.** *As of JDK version 1.1, replaced by* `DateFormat.parse(String s)`.
void	setDate(int date) **Deprecated.** *As of JDK version 1.1, replaced by* `Calendar.set(Calendar.DAY_OF_MONTH, int date)`.
void	setHours(int hours) **Deprecated.** *As of JDK version 1.1, replaced by* `Calendar.set(Calendar.HOUR_OF_DAY, int hours)`.
void	setMinutes(int minutes) **Deprecated.** *As of JDK version 1.1, replaced by* `Calendar.set(Calendar.MINUTE, int minutes)`.
void	setMonth(int month) **Deprecated.** *As of JDK version 1.1, replaced by* `Calendar.set(Calendar.MONTH, int month)`.
void	setSeconds(int seconds) **Deprecated.** *As of JDK version 1.1, replaced by* `Calendar.set(Calendar.SECOND, int seconds)`.

Overview of Date class

- Methods and constructors that are not deprecated
  - getTime() & setTime(long time)
  - Date() & Date(long date)
- A Date object represents a specific instant in time with millisecond precision
  - A Date object contains long value of milliseconds since 1970-01-01 00:00:00.00 GMT (Greenwitch Mean Time)
  - Represents a compact form of maintaining date and time value
  - Much more efficient than *Calendar* object
- Does not maintain timezone or locale information

When Do you want to use Date class?

- Use it when you need to pass a timestamp data efficiently between different parts of your software
- Convert to *Calendar* class when timezone and locale need to be addressed

Calendar & GregorianCalendar Classes

Calendar & GregorianCalendar Classes

- *Calendar* is an abstract base class for converting a moment in time (*Date* object) and a set of integer fields such as YEAR, MONTH, DAY, HOUR, and so on.
  - Subclasses of *Calendar* interpret a timestamp of Date object according to the rules of a specific calendar system.
  - JDK provides one concrete subclass of Calendar: *GregorianCalendar*
- *GregorianCalendar converts* a moment in time (*Date* object) to a particular year, month, day, hour, minute, second etc considering
  - Which timezone to use
  - When the daylight savings begins
  - When the older Julian calendar switched to the Gregorian calendar

GregorianCalendar vs. Date

- *GregorianCalendar* also contains an array of values for holding all of the various fields it knows how calculate.
- All of this additional information as part of its extensive internal state, makes *GregorianCalendar* object many tens of times larger than the much simpler Date object

Different Calendar Systems

- If you needed to calculate the year, month and day and year fields of another calendar you could use a different implementation of *java.util.Calendar*.
  - There are various implementations of calendar available including *ChineseCalendar*, *HebrewCalendar*, and *IslamicCalendar* available

How to create Calendar Object?

- Calendar's *getInstance* static method returns a Calendar object whose time fields have been initialized with the current date and time:
  - *Calendar rightNow = Calendar.getInstance();*
- A Calendar object can produce all the time field values needed to implement the date-time formatting for a particular language and calendar system (for example, Japanese-Gregorian, Japanese-Traditional)

SimpleDateFormat Class

SimpleDateFormat

- *SimpleDateFormat* is a concrete class for formatting and parsing dates in a locale-sensitive manner.
  - It allows for formatting (date -> text), parsing (text -> date), and normalization
- Internally, *SimpleDateFormat* uses a *Calendar* for all date calculations.
  - When you call *Date.toString()*, you are also using a Calendar -- the default calendar -- to calculate date fields.

Properties Class

Properties Class

- The Properties class represents a persistent set of properties
- The Properties can be saved to a stream or loaded from a stream
  - Typically a file
- Each key and its corresponding value in the property list is a string
- A property list can contain another property list as its "defaults"; this second property list is searched if the property key is not found in the original property list

The *Properties* Class: Example

```
// set up new properties object
// from file "myProperties.txt"
FileInputStream propFile
      = new FileInputStream("myProperties.txt");
Properties p
      = new Properties(System.getProperties());
p.load(propFile);

// set the system properties
System.setProperties(p);

// display new properties
System.getProperties().list(System.out);
```

Thank you!

Check JavaPassion.com Codecamps!
http://www.javapassion.com/codecamps
“Learn with Passion!”