Java Utility Classes

Sang Shin
Founder & Chief Instructor
JPassion.com
"Learn with Passion!"

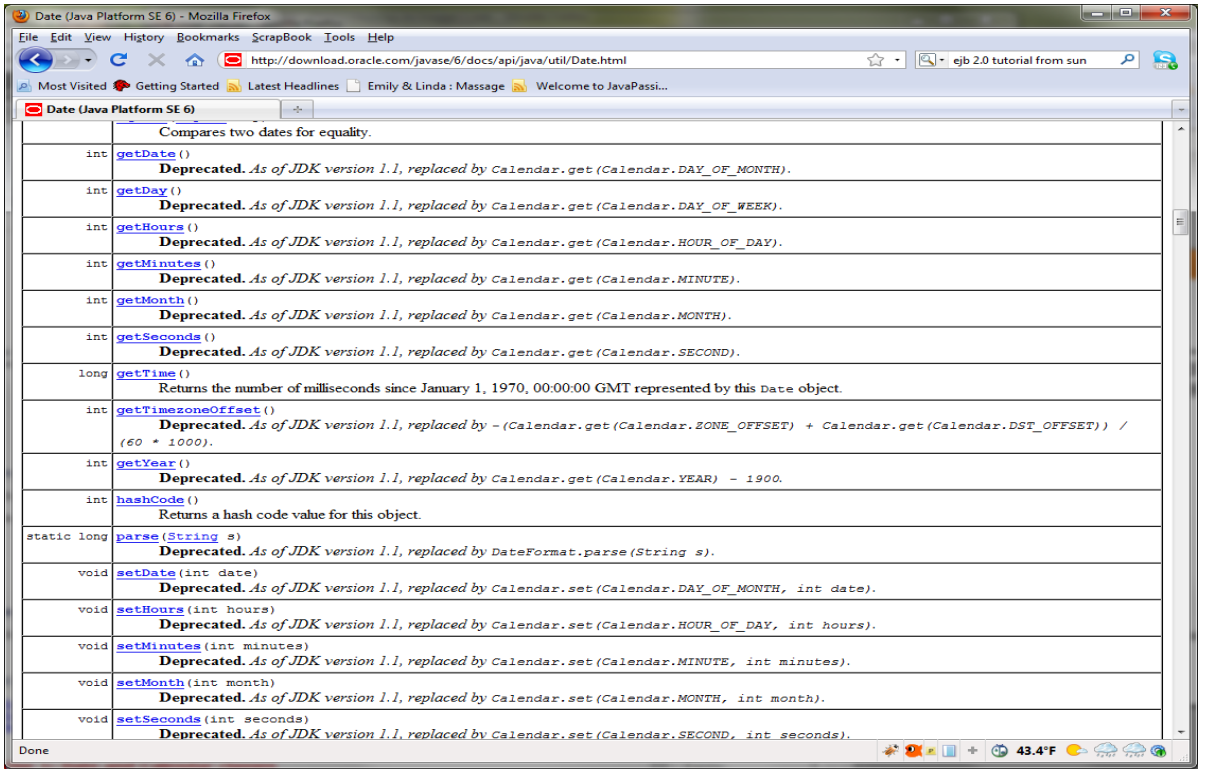
Topics

- Date class
- Calendar and GregorianCalendar classes
- SimpleDateFormat class
- Joda time
- 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)
- http://docs.oracle.com/javase/7/docs/api/java/util/Date.html



Overview of Date class

- Methods and constructors that are not deprecated
  - Methods: getTime() & setTime(long time)
  - Constructors: Date() & Date(long time)
- 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, however

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 information need to be addressed

Lab:

Exercise 1: Date class
1015_javase_utilclasses.zip

Calendar & GregorianCalendar Classes

Calendar & GregorianCalendar Classes

- *Calendar* is an abstract base class for converting a moment in time (*Date* object) to 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 maintains many more information for calculating year, month, day, hour, minute considering timezone, daylight saving time, locale, etc
- 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 other 3rd-party implementations of calendar available including *ChineseCalendar*, *HebrewCalendar*, *BuddistCalendar*, and *IslamicCalendar* available
  - JDK includes only *GregorianCalendar*

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)

Lab:

Exercise 2: Calendar and GregorianCalendar classes

1015_javase_utilclasses.zip

Conversion between Calendar and Date

Conversion between Calendar & Date

- A Calendar object can be created from a Date object and Date object can be created from a Calendar object.

```
// Create a Calendar object from Date object
Date anInstantInTime = new Date();
Calendar c1 = Calendar.getInstance();
c1.setTime(anInstantInTime);

// Create a Date object from a Calendar object
Date utc = c1.getTime();
```

Lab:

Exercise 3: Calendar and Date Conversion
1015_javase_utilclasses.zip

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.

Lab:

Exercise 4: DateFormat & SimpleDateFormat classes
1015_javase_utilclasses.zip

Joda Time

What is Joda Time?

- Joda-Time is a popular 3rd-party library that provides a quality replacement for the Java date and time classes
- The design allows for multiple calendar systems, while still providing a simple API
- The 'default' calendar is the ISO8601 standard which is used by XML. The Gregorian, Julian, Buddhist, Coptic, Ethiopic and Islamic systems are also included

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

```
22          // set up new properties object
23          // from file "myProperties.txt"
24          FileInputStream propFile
25                  = new FileInputStream("myProperties.txt");
26          Properties p
27                  = new Properties(System.getProperties());
28          p.load(propFile);
29
30          // set the system properties
31          System.setProperties(p);
32
33          // display new properties
34          System.getProperties().list(System.out);
```

Lab:

Exercise 5: Properties class
1015_javase_utilclasses.zip

Learn with Passion!
JPassion.com