JSR-310 - A New Date and Time API for Java Platform, Standard Edition (Java SE)

Stephen Colebourne

Technical Architect
SITA ATS Ltd

Michael Nascimento Santos

Senior Technical Consultant
Summa Technologies do Brasil

BOF-2794

https://jsr-310.dev.java.net/

Goal of this BOF

Understand the flaws of the existing Date and Calendar APIs and how JSR-310 intends to tackle them

Agenda

What is wrong with Date and Calendar?
What alternatives are there today?
Why a JSR?
JSR API concepts
Summary
Discussion

Agenda

What is wrong with Date and Calendar?

What alternatives are there today?

Why a JSR?

JSR API concepts

Summary

Discussion

What's wrong with Date?

java.util.Date

- Introduced in JDK 1.0
- Uses two digit years from 1900
- January is 0, December is 11
- Should have been immutable
- Could not be successfully localized
- Most methods deprecated

What's wrong with Calendar?

java.util.Calendar

- Introduced in JDK 1.1
- Designed around untyped int constants
- No constructor to initialize from a Date
- January is 0, December is 11
- Should have been immutable
- Odd performance and bugs
  - Dual internal representation
    - value for each field
    - milliseconds from 1970

What's wrong with SimpleDateFormat?

java.util.SimpleDateFormat

- Requires a Date
  - Calendar cannot be directly formatted
- Not thread-safe
  - Frequent cause of production threading issues
  - Many RFEs raised to make thread-safe

Date example

```
Date date = new Date(2007, 5, 8, 21, 30, 0);

String str = date.toString();
```

Date example – broken

```
Date date = new Date(2007, 5, 8, 21, 30, 0);

String str = date.toString();
```

```
Sat Jun 08 21:30:00 BST 3907
```

Date example – fixed

```
int year = 2007 - 1900;
int month = 5 - 1;
Date date = new Date(year, month, 8, 21, 30, 0);

String str = date.toString();
```

```
Sat May 08 21:30:00 BST 2007
```

Calendar example

```
TimeZone tz = TimeZone.getTimeZone("Asia/Tokyo");
Calendar cal = Calendar.getInstance(tz);

cal.set(1945, Calendar.JULY, 1, 12, 30, 0);

DateFormat f = new SimpleDateFormat("yyyy-MM-dd hh:mm");


String str = f.format(cal);
```

Calendar example – broken

```
TimeZone tz = TimeZone.getTimeZone("Asia/Tokyo");
Calendar cal = Calendar.getInstance(tz);

cal.set(1945, Calendar.JULY, 1, 12, 30, 0);

DateFormat f = new SimpleDateFormat("yyyy-MM-dd hh:mm");


String str = f.format(cal);
```

```
java.lang.IllegalArgumentException:
        Cannot format given Object as a Date
    at java.text.DateFormat.format
    at java.text.Format.format
```

Calendar example – still broken

```
TimeZone tz = TimeZone.getTimeZone("Asia/Tokyo");
Calendar cal = Calendar.getInstance(tz);

cal.set(1945, Calendar.JULY, 1, 12, 30, 0);

DateFormat f = new SimpleDateFormat("yyyy-MM-dd hh:mm");

Date date = cal.getTime();
String str = f.format(date);
```

```
1945-07-02 05:30
```

Calendar example – fixed

```
TimeZone tz = TimeZone.getTimeZone("Asia/Tokyo");
Calendar cal = Calendar.getInstance(tz);

cal.set(1945, Calendar.JULY, 1, 12, 30, 0);

DateFormat f = new SimpleDateFormat("yyyy-MM-dd hh:mm");
f.setTimeZone(cal.getTimeZone());
Date date = cal.getTime();
String str = f.format(date);
```

```
1945-07-02 12:30
```

Agenda

What is wrong with Date and Calendar?

What alternatives are there today?

Why a JSR?

JSR API concepts

Summary

Discussion

Utility libraries

- Company specific or Open source utility libraries
- Simplify some common operations
- No dominant open source utility library
- Still suffer same performance issues

Time and Money

- http://timeandmoney.sourceforge.net
- Open source library
  - MIT license
  - no dependencies
- Wraps the Date and Calendar classes
- Written partly to explore Domain Driven Design
- Still suffers same performance issues

Joda-Time

- http://joda-time.sourceforge.net
- Open source library
  - Apache 2 license
  - no dependencies
- Replacement for Date and Calendar
- Most widely used non Java SE date library
- Mature, version 1.4, developed over 3 years

Agenda

What is wrong with Date and Calendar?

What alternatives are there today?

Why a JSR?

JSR API concepts

Summary

Discussion

Why a JSR?

- Because everyone seemed to want one!

- Existing Date and Calendar APIs are disliked
- Most users of Joda-Time are satisfied
- A 'Joda-Time JSR' kept on being proposed

How will the JSR work?

- JSR-310 is following a very open process
  - public mailing list – anyone can read and write
  - public wiki – anyone can read and write
  - public svn repository – anyone can read
  - public bug and feature request database
- To avoid the mistakes of the past we need input!

https://jsr-310.dev.java.net

Agenda

What is wrong with Date and Calendar?
What alternatives are there today?
Why a JSR?
JSR API concepts
Summary
Discussion

JSR API concepts

- Instant – timeline based
- Human-scale datetimes – field based
  - Fully defined
  - Partially defined
- Interval – between two datetimes
- Duration – scientific definition of time
- Period – such as 3 days, 5 hours and 2 minutes

Instant

- An instant on the timeline
- Defined as a duration from a fixed Epoch
- Used to store a timestamp
- JSR-310 intends
  - to use the standard Java epoch of 1970-01-01
  - to support a range greater than the age of the universe
  - to use nanosecond precision – 10^{-9}

Human-scale datetime

- From complete datetime to single fields
- Defined using human-scale fields
  - year, month, day, hour, minute, second
- Used to store a birthday or shop opening hour
- JSR-310 intends
  - to provide one class per field
  - to provide the most common groups of fields
    - CalendarMonth, CalendarDay, TimeOfDay

Interval

- From a start point to an end-point
- Or, from a start point plus a period
- Used to store conference dates
- JSR-310 intends
  - to provide support for intervals

Duration and Period

- Duration – scientific 'amount of time'
  - measured in seconds
- Period – human-scale
  - such as 3 days, 5 hours and 22 minutes
- JSR-310 intends
  - to provide support for durations and periods

Formatting and Parsing

- SimpleDateFormat is pattern based
- Need more complex date/time formatters/parsers
- JSR-310 intends
  - to provide support for ISO8601
  - to support for SimpleDateFormat style patterns
  - to ensure thread-safety

Open issues

- Leap seconds
- Time zones
- SQL integration
- Calendar systems

Agenda

What is wrong with Date and Calendar?

What alternatives are there today?

Why a JSR?

JSR API concepts

Summary

Discussion

Summary

- Date and Calendar have problems
- Alternatives are available today
  - Joda-Time – http://joda-time.sourceforge.net
- JSR-310 is working to solve this in Java SE

Agenda

What is wrong with Date and Calendar?
What alternatives are there today?
Why a JSR?
JSR API concepts
Summary
Discussion

Open discussion

- What are your burning date and time issues?
- How important is SQL integration?
- How well should we integrate with Date/Calendar?
- Should we include working week/holidays?
- How important are alternate calendar systems?

- Feedback also welcomed on the wiki
  - https://jsr-310.dev.java.net