JSR 310: Uma nova forma de lidar com datas e horas

Fábio Kung
Michael Nascimento Santos

JustJava 2007

Fábio Kung

Michael Nascimento Santos

- 8 anos de experiência com Java
- Co-líder da JSR-310 (Date and Time API)
- Expert nas JSRs 207, 250, 270 (Java 6), 296 (Swing Application Framework), 303 (Bean Validation)
- Co-fundador do SouJava
- Fundador do genesis (https://genesis.dev.java.net) e do ThinNB (https://thinnb.dev.java.net)
- Palestrante no JavaOne, JustJava, Abaporu, FISL, COMDEX, BrasilOne e Conexão Java

Agenda

- Problemas das APIs do JDK
- Alternativas
- Início da JSR
- Decisões, design e conceitos
- Exemplos
- Pontos em aberto
- Como participar

Problemas das APIs do JDK

- java.util.Date
  - ✓ ***Parte do JDK 1.0***
  - ✓ ***Usa dois dígitos para o ano, com base em 1900***
  - ✓ ***Os meses são baseados em zero (Janeiro – 0 ~ Dezembro - 11)***
  - ✓ ***Deveria ser imutável***
  - ✓ ***Não atendia aos requisitos de localização***
  - ✓ ***Quase totalmente deprecated***

Problemas das APIs do JDK

- java.util.Calendar
  - ✓ ***Introduzida no JDK 1.1***
  - ✓ ***Baseada em constantes int, sem tipagem forte***
  - ✓ ***Sem um construtor que receba Date***
  - ✓ ***Os meses são baseados em zero (Janeiro – 0 ~ Dezembro - 11)***
  - ✓ ***Deveria ser imutável***
  - ✓ ***Performance anti-PLA e cheio de bugs***
    - **Representando internamente de duas formas**
      - Um valor por campo
      - Milisegundos a partir de 1970

Problemas das APIs do JDK

- java.text.SimpleDateFormat
  - ✓ ***Requer um Date***
    - **Não é possível formatar diretamente um Calendar**
  - ✓ ***Não é thread-safe***
    - **Diversos problemas difíceis de reproduzir podem ocorrer em produção**
    - **Vários RFEs presentes no banco de bugs do JDK para torná-la thread-safe**

Problemas das APIs do JDK

- java.util.TimeZone
  - ✓ ***Regras "hardcoded"***
  - ✓ ***Não possui acesso às regras***
  - ✓ ***Problemas quando regras são incompatíveis para um mesmo TimeZone em duas JVMs***
  - ✓ ***Não permite especificar regras mais complexas presentes no mundo real nem mudanças nas regras (Brasil / Venezuela)***

Problemas das APIs do JDK

- TimeZone implícito
- Sem API para datas sem hora e horas
- Sem API para representar durações, ex: 3 dias
- Sem API para representar períodos, i.e., 05/10/2007 13:00~14:00
- Formato de serialização inapropriado
- Resultados inesperados na serialização entre JVMs distintas
- Dificuldade de trabalhar com outros sistemas de calendário

Exemplo: Date

```
Date date = new Date(2007, 10, 5, 13, 0, 0);
System.out.println(date.toString());
```

Exemplo: Date

```
Date date = new Date(2007, 10, 5, 13, 0, 0);
System.out.println(date.toString());
```

```
Tue Nov 05 13:00:00 GMT-03:00 3907
```

Exemplo: Date

```
int ano = 2007 - 1900;
int mes = 10 - 1;

Date date = new Date(ano, mes, 5, 13, 0, 0);
System.out.println(date.toString());
```

```
Fri Oct 05 13:00:00 GMT-03:00 2007
```

Exemplo: Calendar / DateFormat

```
TimeZone tz = TimeZone.getTimeZone("America/Manaus");
Calendar cal = Calendar.getInstance(tz);

cal.set(2007, Calendar.OCTOBER, 5, 13, 0, 0);

DateFormat f = new SimpleDateFormat("dd/MM/yyyy HH:mm");


System.out.println(f.format(cal));
```

Exemplo: Calendar / DateFormat

```
TimeZone tz = TimeZone.getTimeZone("America/Manaus");
Calendar cal = Calendar.getInstance(tz);

cal.set(2007, Calendar.OCTOBER, 5, 13, 0, 0);

DateFormat f = new SimpleDateFormat("dd/MM/yyyy HH:mm");


System.out.println(f.format(cal));
```

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

Exemplo: Calendar / DateFormat

```
TimeZone tz = TimeZone.getTimeZone("America/Manaus");
Calendar cal = Calendar.getInstance(tz);

cal.set(2007, Calendar.OCTOBER, 5, 13, 0, 0);

DateFormat f = new SimpleDateFormat("dd/MM/yyyy HH:mm");

Date date = cal.getTime();
System.out.println(f.format(date));
```

```
05/10/2007 14:00
```

Exemplo: Calendar / DateFormat

```
TimeZone tz = TimeZone.getTimeZone("America/Manaus");
Calendar cal = Calendar.getInstance(tz);

cal.set(2007, Calendar.OCTOBER, 5, 13, 0, 0);

DateFormat f = new SimpleDateFormat("dd/MM/yyyy HH:mm");
f.setTimeZone(cal.getTimeZone());
Date date = cal.getTime();
System.out.println(f.format(date));
```

```
05/10/2007 13:00
```

Alternativas hoje

- Bibliotecas utilitárias
  - ✓ ***Feitas para a empresa/projeto***
    - **Difíceis de desenvolver, repletas de bugs etc.**
  - ✓ ***Time and Money (http://timeandmoney.sourceforge.net)***
    - **Continua trabalhando em cima de Date e Calendar**
    - **Sofre dos mesmos problemas de performance e serialização**
  - ✓ ***Joda-Time (http://joda-time.sourceforge.net)***
    - **Substitui completamente Date e Calendar**
    - **Estável, existe há 3 anos**
    - **Opção mais recomendada**

JSR-310: Início

- Usuários do Joda-Time que apóiam padrões pediam constantemente uma JSR
- No 2o semestre/2006, finalmente foi decidido iniciar a JSR
- No início de 2007, foi submetida por Michael Nascimento Santos e Stephen Colebourne, com o apoio de 17 membros do JCP
- Aprovada com 16 votos sim, zero não e 1 abstenção
- Conduzida em público - https://jsr-310.dev.java.net/

Decisões

- Não usa diretamente código do Joda-Time
  - ✓ ***Há conceitos muito específicos***
  - ✓ ***Não temos o copyright de todo o código***
- Compatibilidade com Java 5
  - ✓ ***Pode mudar para Java 7 se houver operator overloading***
- Suporte ao padrão ISO-8601
- Precisão de nanos, ao invés de milis
- Sustitui API antiga, mantendo interoperabilidade sempre que possível

Princípios de design

- Imutabilidade
- Fluent API
- Clara, explícita e age como esperado (PLA)
- Extensível

Conceitos

- Instants/timestamps: pontos únicos na linha do tempo
- Human-scale datetimes/Timepoints: baseados em campos, completos (podem ser convertidos para instantes) ou incompletos
- Interval: um intervalo aberto entre dois instants
- Duration (antes Period): representa o conceito de 1 semana, 2 dias, 3 horas
- Recurrent time: toda segunda quarta-feira do mês
- Calendar systems & timezones

Outros aspectos a serem considerados

- Formatação
- Parsing
- Serialização
- Atualização das informações de TimeZone
- Horário da JVM
- Interoperabilidade com a API antiga
- Proposta de atualização de APIs existentes (dependem das respectivas JSRs):
  - ✓ ***JDBC***
  - ✓ ***JPA***

Exemplo: Construindo datas

```
CalendarDay date = calendarDay(2007, 3, 20);
date = calendarDay(year(2007), march(),
  dayOfMonth(20));
date = calendar().year(2007).december().dayOfMonth(20)
  .buildLenient();
date = calendar().year(1972).december().dayOfMonth(3)
  .build();
date =
  calendar().currentYear().december().dayOfMonth(20)
  .buildLenient();
date = calendar().zoneID("America/Manaus").year(2007)
  .august().dayOfMonth(2).build();
```

Exemplo: Manipulando datas e horas

```
TimeOfDay tod = currentTime();
tod = tod.plusHours(6).plusMinutes(2);
tod = tod.plus(hours(6), minutes(2));
// com operator overloading:
// tod += hours(6);

CalendarDay date = today();
date = today().plusDays(3);
date = today().plus(days(3));
date = now().today().plus(Days.days(3));
// com operator overloading:
// date += days(3);
// date += 3;
```

Pontos em aberto

- Grau de precisão das classes
  - ✓ ***Tipos para tudo***
  - ✓ ***Classe genérica parametrizável***
  - ✓ ***Enums sempre que possível***
- Integração com a JSR-275 (Units)
- Suporte a leap seconds
- Sistemas de calendário a serem suportados
- Grau de padronização do acesso e atualização das classes de TimeZone
- Interoperabilidade com XMLGregorianCalendar (limitaria as opções de design)

Como participar

- Tudo é público em:
  - ✓ ***https://jsr-310.dev.java.net/***
- Implementação de referência no repositório Subversion
- Todas as questões de design, princípios, conceitos, pontos em aberto, propostas, casos de uso no Wiki
- Lista pública de discussão no site do projeto
- Até o momento, nenhuma discussão ou decisão ocorreu em particular

Perguntas?

Obrigado!

https://jsr-310.dev.java.net/
http://blog.michaelnascimento.com.br/

Michael Nascimento - michael@summa-tech.com
Fábio Kung

JustJava 2007