Go for the money! JSR-354

A quick introduction to the Java Money and Currency API

Marcus Fihlon

July 7, 2016

Scrum Master | Software Engineer | Lecturer | Speaker

DISCLAIMER

The following presentation has been approved for open audiences only. Hypersensitivity to occasional profanity requires covering ears.

All logos, photos etc. used in this presentation are the property of their respective copyright owners and are used here for educational purposes only. Any and all marks used throughout this presentation are trademarks of their respective owners.

The presenter is not acting on behalf of CSS Insurance, neither as an official agent nor representative. The views expressed are those solely of the presenter.

Marcus Fihlon disclaims all responsibility for any loss or damage which any person may suffer from reliance on this information or any opinion, conclusion or recommendation in this presentation whether the loss or damage is caused by any fault or negligence on the part of presenter or otherwise.

NOTES AND PHOTOS

You can take notes and photos if you want.
Or you can focus on the presentation and live coding.

All my slides and source files are available online.
A link and a QR code are on the last slide.

ABOUT ME

- **Scrum Master**
  CSS Insurance
- **Software Engineer**
  CSS Insurance / Open Source Software
- **Lecturer**
  TEKO Swiss Technical College
- **Speaker**
  Conferences / User Groups / Meetups

www.fihlon.ch | github.com/McPringle | hackergarten.net

AGENDA

Intro

API

Live Coding

Wrap-up

INTRO

MONEY

A large proportion of the computers in this world manipulate money, so it's always puzzled me that money isn't actually a first class data type in any mainstream programming language. The lack of a type causes problems, the most obvious surrounding currencies...

Martin Fowler

BEFORE

- `float`, `double` (since Java 1)
- `java.math.BigDecimal` (since Java 1.1)
- `java.text.DecimalFormat` (since Java 1.1)
- `java.util.Currency` (since Java 1.4)

MOTIVATION

- Monetary values are a key feature for many applications
- `java.util.Currency` is a structure for ISO-4217 only
- No standard value type to represent a monetary amount
- No support for currency arithmetic or conversion
- No support for historic or virtual currencies
- `java.text.DecimalFormat` lacks flexibility

REQUIREMENTS

- Easy **addition** and **modification** of currencies
- Currencies need a **context** and should be **client-aware**
- Standard API for **money amounts**, **rounding**, **conversions**
- Easy, flexible, complex, individual **formatting** and **parsing**
- Clearly defined **extension points**
- Follow the **design principles** of the Java platform
- Compatibility with **Standard Edition** and **Micro Edition**
- No external **dependencies**
- **Interoperability** with existing artifacts
- Support **functional** programming style

FOUR YEARS OF HARD WORK

- JSR-354 started early 2012
- JSR-354 early draft review in 2013
- Reference implementation startet late 2013
- Final release of JSR-354 early 2015
- Final release of reference implementation early 2015

API

CURRENCIES

- `Monetary`
  - Get a currency by code or locale
  - Additional API for complex queries
  - Supports SPI to enhance functionality
- `CurrencyUnit`
  - Currency code (string and number)
  - Additional context information (e.g. type, capabilities)
- `CurrencyQueryBuilder`
  - Builder for complex queries for accesing currency units

MONETARY AMOUNTS

- `Monetary`
  - Get a monetary amount by currency and value
  - Optionally specify an explicit factory
  - Or query for a suitable factory
  - Supports SPI to enhance functionality
- `MonetaryAmount`
  - Numeric value and currency
  - Arithmetic operations to do calculations
  - Multiple implementations
  - Interoperability rules
  - Additional context information (e.g. capabilities)
- `MonetaryAmountFactoryQueryBuilder`
  - Builder for complex queries for accesing monetary amout factories

ROUNDING

- `Monetary`
  - Get a rounding operator
  - Optionally specify a locale
  - Or query for a suitable rounding operator
  - Supports SPI to enhance functionality
- `MonetaryRounding`
  - Extends `MonetaryOperator`
  - Multiple implementations
  - Additional context information
- `RoundingQueryBuilder`
  - Builder for complex queries for accesing rounding operators

Conversion with currency conversion

- `MonetaryConversions`
  - Get a currency conversion by currency code or unit
  - Or query for a suitable currency conversion
  - Supports SPI to enhance functionality
- `CurrencyConversion`
  - Multiple implementations
  - Source and target currency
  - Conversion factor
  - Additional context information
  - Unidirectional
- `ConversionQueryBuilder`
  - Builder for complex queries for accesing currency conversions

Conversion with exchange rate provider

- `MonetaryConversions`
  - Get an exchange rate provider
  - Or query for a suitable currency conversion
  - Supports SPI to enhance functionality
- `ExchangeRateProvider` $\rightarrow$ `ExchangeRate`
  - Multiple implementations
  - Source and target currency
  - Conversion factor
  - Additional context information
  - Unidirectional
- `ConversionQueryBuilder`
  - Builder for complex queries for accesing exchange rate providers

- `MonetaryFormats`
  - Get a monetary amount format
  - Optionally specify a locale
  - Or query for a suitable monetary amount format
  - Supports SPI to enhance functionality
- `MonetaryAmountFormat`
  - Multiple implementations
  - Additional context information
  - Format monetary amounts
  - Parse monetary amounts
- `AmountFormatQueryBuilder`
  - Builder for complex queries for accesing monetary amount formats

LIVE CODING

Live Coding

The code of the demo will be available on GitHub.

WRAP-UP

INTERFACES

- CurrencyConversion
- CurrencySupplier
- CurrencyUnit
- ExchangeRate
- ExchangeRateProvider
- ExchangeRateProviderSupplier
- MonetaryAmount
- MonetaryAmountFormat
- MonetaryOperator
- MonetaryQuery
- MonetaryRounding
- NumberSupplier

SERVICE PROVIDER INTERFACES

- CurrencyProviderSpi
- MonetaryAmountFactoryProviderSpi
- MonetaryAmountFormatProviderSpi
- MonetaryAmountsSingletonQuerySpi
- MonetaryAmountsSingletonSpi
- MonetaryConversionsSingletonSpi
- MonetaryCurrenciesSingletonSpi
- MonetaryFormatsSingletonSpi
- MonetaryRoundingsSingletonSpi
- RoundingProviderSpi

Money Validation by Zalando

- Validate monetary amounts
- Uses existing, standardized constraints
- Offers additional, more expressive custom constraints
- Can be use with any Bean Validation implementation

```
1 @Min
2 @Max
3 @DecimalMin
4 @DecimalMax
5 @Positive
6 @PositiveOrZero
7 @Negative
8 @NegativeOrZero
9 @Zero
```

COMPATIBILITY

- Supports
  - Java Micro Edition
  - Java Standard Edition
  - Java Enterprise Edition
- Compatible with Java 8+
- Backport available for Java 7

pom.xml

```
<dependency>
    <groupId>javax.money</groupId>
    <artifactId>money-api</artifactId>
    <version>1.0.1</version>
</dependency>
<dependency>
    <groupId>org.javamoney</groupId>
    <artifactId>moneta</artifactId>
    <version>1.1</version>
</dependency>
```

build.gradle

```
compile(
    'javax.money:money-api:1.0.1',
    'org.javamoney:moneta:1.1'
)
```

LINKS

- Java Money Umbrella Site
  `http://javamoney.org/`
- JSR-354 Specification
  `http://javamoney.github.io/api.html`
- JSR-354 Reference Implementation
  `http://javamoney.github.io/ri.html`
- JSR-354 Technical Compatibility Kit
  `http://javamoney.github.io/tck.html`
- Java Money Financial Library
  `http://javamoney.github.io/lib.html`
- Money Validation
  `https://github.com/zalando/money-validation`

Thank You! Questions?

http://bit.ly/jsr-354