Javaの現在と未来（Ver.5~8）

株式会社アットウェア　佐竹 雅央
atWare, Inc. SATAKE Masahiro

じこしょうかい

- 仕事歴
  - 2002年に就職して以来、ほぼJava一筋
  - もっぱらWebアプリケーション
- 函館歴
  - 今回が2回目です
  - 前回は去年の5月
    - アットウェア函館ラボ開所記念セミナ
      - 「JavaOne2008報告」

Javaのバージョニング

- Java Platform 1.1.X_YYY
  - 1.1.1_001
- J2SE 1.2.X_YYY
  - J2SE はJava 2 Platform Standard Editionのこと
  - 1.2.2_013 からは 1.2.2_13/013 と2桁併記
- J2SE 1.3.X_YY
- J2SE 1.4.X_YY
- J2SE 5.X Update Y
  - 略して 5.XuY
  - Xは0だけだった
- Java SE 6 Update Y
  - 略して 6uY

J2SE 5.0

- 多数の言語仕様追加が行われた
  - コンセプトは「EoD（Easy of Development）」
    - 小技：シンタックスシュガー（構文糖）
      - 面倒な頻出パターンを文法レベルでサポート
      - 短くシンプルに書ければ生産性向上、ケアレスミスも減る
    - 大技：タイプセーフの追求
      - 実行時にバグとなりうるコードはコンパイルを通さない
    - 必殺技：注釈構文（アノテーション）による自動化
      - ルールに従ったアノテーションを書いておけば
        - » コンパイラがチェックしてくれる
        - » ジェネレータが残りを作ってくれる
        - » 実行時にライブラリ・フレームワークが何かしてくれる

J2SE 5.0

- 『シンタックスシュガー』
  - Autoboxing/Unboxing
    - プリミティブ型とオブジェクト型（ラッパークラス）の自動変換

```
Integer one = 1;
```

```
int  two = new Integer(2);
```

  - Enhanced for Loop（The For-Each Loop）
    - 配列とコレクションの要素繰り返しのための簡易For構文

```
for (String str : strings) { System.out.println(str);}
```

  - Static Import
    - メンバまで指定したimport（staticなメンバに限る）
      - 定数やユーティリティ的なメソッドの利用に
  - Varargs
    - 可変長引数

```
public void method(String… args) {
  System.out.println(args.length);
}
```

    - メソッド内では配列扱い

J2SE 5.0

- シンタックスシュガーの甘い罠（の例）
  - NullPointerExceptionが発生
  - 発生行は `hoge = piyo.fuga();`
    - 今までの常識から言って、原因はpiyoがnullでFA
    - しかし、いくらデバッグしてもpiyoがnullにはならない
      - デプロイに失敗してるんじゃない？
      - デバッガが副作用を引き起こしているんじゃないか？
      - JavaVM壊れた？
  - 実は、hoge は int で piyo.fuga() は Integer

```
hoge = piyo.fuga().intValue();
       ^^^^^こっちが null
```

J2SE 5.0

- 『タイプセーフの追求』
  - Generics
    - 汎用ライブラリを非汎用に使う「型パラメータ」
      - 何でも入れられる＝何が入っているか分からない
      - 敢えて入れるものを制限する＝安心して取り出せる

```
List<String> list = new ArrayList<String>();
list.add(new Integer(1)); // コンパイルエラー
String element = list.get(0); // キャスト不要
```

      - 宣言の側はこんな感じ

```
Interface List<E> extends Collection<E> {
  boolean add(E o);
  E get(int index);
}
```

J2SE 5.0

- 『タイプセーフの追求』
  - Typesafe Enums
    - 型保証された列挙型
      - 同名デザインパターンを言語レベルでサポート
        - » 新キーワード enum で、特殊なクラスとして定義

```
enum Camp {GI, GO , SHOKU};
enum Permission{READ, WRITE, EXEC};

Camp camp = Permission.READ;
```

      - 定数による列挙型の良さも失わないよう
        - » switch文を拡張してint型に加えEnumも指定可能に
        - » ビット演算の代用にできるEnumSetを用意

```
switch (camp) {
  case GI:
  case GO:
  case SHOKU
}
EnumSet set =   EnumSet.of(Permission.READ, Permission.WRITE);
```

J2SE 5.0

- 『注釈構文（アノテーション）による自動化』
  - Annotation
    - フレームワークなどに対して、「設定ファイル」「命名規約」以外の方法で、簡単かつ明示的に情報を渡せるようになった
      - 「設定ファイル」の問題点
        - » 大規模化に伴うXML地獄
        - » Javaソースと設定ファイルの配置的距離・二重管理
      - 「命名規約」の問題点
        - » スペルミスなどで処理対象外になっても気付かず
        - » 「フレームワークのための命名規約」≠「本質的な名前」
    - フレームワーク開発者がより柔軟で扱いやすい仕組みを提供しやすくなった
    - コンパイラでの妥当性チェックにも使用される
      - 実行時にバグとなりうるコードはコンパイルを通さない、という考え方

```
@Foo(hoge="xxx",piyo="ooo")
protected Object clone() {…}
```

```
@Overwrite
protected Object clone() {…}
```

J2SE 5.0

- 注釈保持ポリシーと保持範囲

注釈保持ポリシー	注釈の保持範囲	利用例
SOURCE	ソースファイル上にのみ存在する。コンパイラがバイトコード化する際には無視される。	・XDoclet等のジェネレータ ・APT（Annotation Processing Tool） アノテーション向けプリプロセッサ ・コンパイラ
CLASS	クラスファイル（バイトコード）まで残る。JVMがロードする際には無視される。	・IDE（統合開発環境） ・コンパイラ ・バイトコード変換ライブラリ
RUNTIME	JVMがロードした後も情報が残り、リフレクションを使って実行プログラム内から参照できる。	・フレームワーク ・ライブラリ ・Javaプログラム全般

  - 注釈の注釈、メタアノテーションで示される

J2SE 5.0

- 追加された標準APIの紹介（個人的choiceによる）
  - java.lang.instrument
    - クラスのロード直前にバイトコード変換プログラムを呼び出す、橋渡しの処理を書くためのAPI
      - instrument：計測 のパッケージ名で元々は監視やデバッグのためのバイトコード変換だが、それ以外の目的でも問題ない
      - javaコマンドのオプション「-javaagent」と組み合わせて使う
  - java.util.concurrent
    - 並列処理のより強力なサポート
      - アトミック・ロック・スレッドセーフコレクション（＋キュー）
  - java.lang.management / javax.management
    - JMX（Java Management eXtensions）
      - VM内の情報をVM外から監視・管理させる仕組み

J2SE 5.0

- 実行環境(jre)向けユーティリティ
  - CDS（Class Data Sharing）
    - JVM間で標準クラスを共有しメモリ消費の軽減と起動時間の短縮
    - Windows限定機能
- 開発環境(jdk)向けユーティリティ
  - 監視と管理のためのツール
    - jconsole, jps, jstat/jstatd
    - 一部を除きLinux限定
  - トラブルシューティングツール
    - jinfo, jmap, jsadebugd, jstack
    - 一部を除きLinux限定
  - APT(Annotaion Processing Tool)
    - ソースファイルに記述されたアノテーションを使って、元のソースの内容を書き換えたり新しいソースファイルを追加したりするためのJavaライブラリ（標準APIではない、sunパッケージ）と、それを呼び出すためのコマンドツール。

※これらはsunのJRE/JDKに付随するツールなので、他のベンダーが作ったJRE/JDKには無いかもしれない

JavaSE 6

- 新しい言語仕様の追加は「なし」
  - コンセプトは引き続き「EoD」
  - 5.0で取り入れられたEoDな仕組みを、標準API自身に、より十分に浸透させる努力が行われた
    - アノテーションの追加や微調整
    - Genericsへの対応（Parameterize:パラメータ化）
    - 定数をEnumに置き換え（または併用）
    - 可変長引数の利用

JavaSE 6

- 追加された標準APIの紹介（個人的choiceによる）
  - javax.annotation.processing/javax.lang.model
    - APTが標準APIに進化
      - sunのJDK独自のsunミラーAPIを使用せずに済む
      - javacコマンドのオプション「-processor」と組み合わせて使う
  - javax.script
    - JVM上で動くスクリプト言語を、Javaプログラムから実行できる
  - javax.xml.transform.stax/javax.xml.stream
    - 第3のXMLパーサ StAX（Streaming API for XML）
      - SAX（Simple API for XML）がpushならStAXはpull
        - » SAXは先頭からシーケンシャルにイベントが飛んでくるので、自作したハンドラでそれらを受け止める
        - » StAXは基本的にこちらから「次をよこせ」と言う
        - » Filterを使って読み飛ばしたり、途中で「もういいや」も可
    - javax.xml.transform.stream はちょっと別物なので紛らわしい。。。
  - javax.xml.bind
    - JAXB（Java Architecture for XML Binding）
    - O/Rマッパーの XML版・・・のような感じ

突然ですが JavaFX

- 広義には
  - Javaをコンシューマサイドで活躍させるための新しいプラットフォーム（実行環境＋開発環境＋α）
    - FlashなどのRIAに対抗
- 狭義には
  - JavaFX Script
    - JVM上で実行できる、GUIに特化したスクリプト言語
    - 普通のJavaで作成したクラスも簡単に連携できる
    - 目下のところは
      - アプレットとしてブラウザ上で動かす（JavaFX Desktop）
      - Javaをサポートする携帯アプリとして（JavaFX Mobile）

JavaSE 6u10

- JavaFX 普及大作戦？ JREの機能を大幅強化
  - Javaカーネル（Windows限定）（旧名 Java Browser Edition）
    - jarファイルのダウンロードとクラスのロードを並列実行
    - 先に必要なjarを先にダウンロード
  - Quick Starter（Windows限定）
    - CDS専用のJVM（のようなもの）を事前に起動しておくことで、新しくJVMを起動する際のパフォーマンスを向上
  - Deployment Toolkit
    - JavaFXScriptをアプレットとして動かすために必要なHTMLタグを動的に出力するJavaScriptライブラリ
    - WEBページへのJavaFX配備が簡単になった
  - 次世代Java Plug-in
    - ブラウザのプロセス上で作成していたJVMを、ブラウザから独立して起動するようになった
    - ブラウザで取得/表示したアプレットをそのままブラウザから切り離してデスクトップアプリケーションとして使用可能

Java年表

- 1995 Javaを発表（Java 1.0β）
- 1996 Java 1.0(JDK1.0)
- 1997 Java 1.1(JDK1.1)
- 1999 J2SE 1.2(J2SDK1.2)
- 2000 J2SE 1.3(J2SDK1.3)
- 2002 J2SE 1.4(J2SDK1.4)
- 2004 J2SE 5.0(JDK5.0)
- 2006 JavaSE 6(JDK6)

Javaの「現在」

- JavaSE 7 が・・・ でそうでででない
  - この間までは2009年夏予定だった気がする
    - 来年の春以降までオアズケになりました
      - その前は2008秋って予定もあったんだけどね
  - 遅れた原因は？
    - JavaFXで頭が一杯だった
    - JavaFXのためのjre6u10作りで手が一杯だった
    - OpenJDKとかやっちゃったし
    - 景気悪いし?

JavaSE 7

- 幾つもの「言語仕様」の変更が検討されてきた
  - 小技
    - Multi catch (ひとつのcatch節で複数種類の例外をキャッチ)
    - Safe rethrow（catch節の仮引数をfinal宣言すると・・・）
    - nullデリファレンス式(?を使ってnullチェック&ハンドリング)
    - 型推論による実型パラメータの省略(List<String> list = new ArrayList<>();)
  - 大技
    - プロパティ(property キーワードでアクセッサ省略)
    - モジュール化(JARからJAMへ)
    - スーパーパッケージ(真の親子関係を)
    - XMLリテラル
    - 型宣言に対するアノテーション(ヌルが入れられない変数、など）
    - BigDecimalの四則演算子対応
    - 演算子のオーバーロード(ただし限定的)
  - 必殺技
    - クロージャ

JavaSE 7

- 幾つもの「言語仕様」の変更が検討されつづけている
  - 小技
    - Multi catch (ひとつのcatch節で複数種類の例外をキャッチ)
    - Safe rethrow（catch節の仮引数をfinal宣言すると・・・）
    - nullデリファレンス式(?を使ってnullチェック&ハンドリング)
    - 型推論による実型パラメータの省略(List<String> list = new ArrayList<>();)
  - 大技
    - プロパティ(property キーワードでアクセッサ省略) →7には間に合わなさそう
    - モジュール化(JARからJAMへ) →7には間に合わなさそう
    - スーパーパッケージ(真の親子関係を) →形を変えて採用？「Project Jigsaw」
    - XMLリテラル →7には間に合わなさそう（っていうかやめた？）
    - 型宣言に対するアノテーション(ヌルが入れられない変数、など）
    - BigDecimalの四則演算子対応 →7には間に合わなさそう
    - 演算子のオーバーロード →7には間に合わなさそう
  - 必殺技
    - クロージャ →7には間に合わなさそう→ただし、根強い人気が・・・

JavaSE 7

- 追加されそうな標準API
  - New I/O 2（ツー）
    - 完全非同期I/Oやリアルタイムファイルシステム API
  - 並列処理の更なる強化
  - Swingアプリケーションフレームワーク
    - これもJavaFX Scriptを意識してのこと？
      - 内容未確認（・・;

JavaSE 7

- クロージャは本当に必殺技か？
  - そのほかの大技も、本当にJavaに必要なのか？
    - 今ある文法でも殆どの場合対応できる
    - あれもこれもと取り込んで、ぐちゃぐちゃに・・・
    - 変化しなければすぐに死んだ言語になってしまう
    - Java開発者がJavaもやるじゃん、と思えることが大事
      - 意見の相違が激しいクロージャだが、取り入れてほしいと思われてるランキングではダントツ。

Javaの「未来」

- 政治的に
  - 買収されんのか
  - ASFと仲直りは
- 言語として
  - いつまでも技術者に好かれ続けるのか
  - そろそろ愛想を尽かされるのか
- プラットフォームとして
  - コンシューマサイドで復活できるのか
  - 他言語多言語のためのプラットフォームになるか

おわり

- おそまつさまでした