JIM LASKEY

JDK 9のJavaScriptエンジン Nashorn

便利な追加機能とES6のサポートでNashornがさらに充実

今回の新リリースでは、JDKの組込みJavaScriptエンジンであるNashornにもさまざまな改善が施されています。まずは、機能拡張について確認する前に、その背景について簡潔に説明するところから始めます。そうすることによって、なぜ今回の変更が行われたのか、納得していただけるはずです。

背景

筆者がNashornの開発を始めたのは2010年後半であり、新しく追加されたinvokedynamicバイトコード命令（JSR 292）の実験を行う方法を探していたときのことでした。その後、JVMチームはinvokedynamicの試験台としてNashornを採用し、Nashornプロジェクトはinvokedynamicの実装のために大幅なパフォーマンス改善を行いました。

それと並行して、Javaグループでは、クライアントサイドの開発でどの程度JavaScriptが優勢になる可能性があるのかについてや、クライアントでもサーバーでもJavaScriptとJavaの統合が重要な要素になることが話し合われていました。当時のJDKがJavaScript向けに提供していたのは、Mozilla Foundationによるオープンソース実装のRhinoでした。Rhinoは古くなり始めており、開発は縮小される方向に向かっていました。そして、2012年11月にJDK拡張提案（JEP）174「Nashorn JavaScriptエンジン」が承認されました。この承認によって、Javaプラットフォームで実行され、強固かつ安全ですべての機能を搭載した、ECMAScript-262 Edition 5.1（ES5）の新たな実装の提供に向けて、本格的な作業を始めることが可能になりました。

当初、Nashornは、アプリケーション・サーバー、JavaFXアプリケーション、ユーティリティ、シェル・スクリプト、埋込みシステムなどさまざまなアプリケーションに使われていました。

Nashornは現在も広く使われているものの、その用途は主に3つの領域に落ち着いているように見受けられます。

- クライアントとサーバーの両方で実行できるJavaScriptアプリケーションの開発：これはJavaScriptの世界でアイソモーフィック開発と呼ばれる方式で、デスクトップとモバイルの両方でサービスを提供するフロントエンドを構築したい小規模な店舗に多大なメリットをもたらします。共通のコードベースを持つ1つのプログラミング言語を使って短時間で開発できるからです。アイソモーフィック開発は、大規模システムにも拡張できます。
- ランタイム適応コーディング（ダイナミック・コーディング）：筆者が好んで使う用語は、ソフト・コーディングです。これは、アプリケーションやサーバーのデプロイ後にもアプリケーションの各部を変更できることを意味し、データベースのストアド・プロシージャからアプリケーションの構成管理まで、あらゆることに使われています。
- シェル・スクリプト：従来はbashやPythonが使われていた領域でJavaScriptを使うことです。

GoogleのV8などのプラットフォームで実行されるJavaScriptネイティブのパフォーマンスと比べた場合の、JVM上で実行されるNashornのパフォーマンスについては、多くの議論が交わされています。NashornはJavaScriptをバイトコードに変換しているため、動作の開始が遅くなります。その後のNashornのパフォーマンスは、ネイティブ・コードを実行するHotSpot次第です。このアプローチは、長時間にわたって実行を続けるサーバー・アプリケーションには向いていますが、一度だけ実行される小さなスクリプトにはあまり適していません。そもそも、Nashornが開発された主な理由は、JavaScriptとJavaをシームレスに同時実行することでした。ほとんどのNashornアプリケーションは、JavaScriptとJavaを協調させるという仕事をうまくこなしています。V8はこのような目的で設計されたものではありません。

ES6のサポート

アイソモーフィック開発を行うJavaScript開発者にとってもっとも重要な問題は、クライアントとサーバーのソース・コードに互換性を持たせる必要があることです。ほとんどのブラウザがJavaScript言語サポートの標準レベルとしてECMAScript 6 (ES6) を採用しているため、NashornにもES6をサポートさせることは必須でした。Nashornチームには、JDK 9リリースで完全なES6機能セットを提供するだけの時間はありませんでしたが、今後の更新では、さらに多くのES6機能がサポートされる予定です。ES6サポートを有効化するためには、コマンドラインで--language=es6を指定して下さい。

新しいキーワード：最初に気づくのは、新しいキーワードとして、ES6のブロック・スコープに従うletとconstが実装されていることです。constは、不変ローカル変数を作成します。下位互換性のため、varはES5.1のセマンティクスを維持しています。

```
const a = 10;
a = 20; // 定数に設定しようとしているため、TypeError

let x = 10;
{
   let x = 20; // スコープが異なる
   print(x); // 20を出力
}
print(x); // 外側のスコープの10を出力
```

シンボルのサポート：シンボルは、ES6で導入された、文字列に似たオブジェクトです。シンボルはプリミティブですが、値ではなく参照を使って比較されます。つまり、すべてのシンボルが一意になるため、開発者がプライベートなオブジェクト・プロパティを作成できるようになります。

```
// 新しい一意なシンボル
let unique = Symbol('optionalName');
myobj[unique] = 'foo';
// 指定した名前の共有シンボル
let shared = Symbol.for('name');
myobj[shared] = 3;
```

最初の例では、個別のシンボルを作成しています。これは、現在のスコープに対してプライベートなプロパティを作成するために使うことができます。2番目の例は、コンテキスト全体でシンボルを共有できるようにする (internする) 方法を示しています。ES6では、iteratorとデフォルトのtoString関数の定義に共有シンボルが使われています。シンボルをキーとしたプロパティは、リフレクション操作を使っても参照することはできません。

新しいiteratorプロトコル：ES6では、オブジェクトに対して反復処理を行う新しいプロトコルが提供されています。

```
// Symbol.iteratorプロパティに
// iterator関数を割当て
myobj[Symbol.iterator] = function() {
   return {
      next: function() { ... }
   }
};
// for..of文を使ってmyobjに対して反復処理を実行
for (let id of myobj) {
   print(myobj[id]);
}
```

for..ofループが導入されていることに注意してください。Nashornのfor each構文もまだ使える見込みですが、同じ機能であっても、for..ofに切り替えることをNashornチームは推奨しています。

新しいコレクション：ES6およびNashornには、新しいコレクション・クラスであるMapとSetが追加されています。この2つのコレクション・クラスには、新しいiteratorプロトコルが実装されています。

```
let map = new Map();
map.set('foo', 'bar');
map.get('foo');  // -> 'bar'
map.clear();
map.get('foo');  // -> 未定義
```

MapとSetの弱参照バージョンであるWeakMapとWeakSetも実装さ

れています。この2つのコレクションでは、エントリ値が他のどの変数やオブジェクトからも参照されなくなると、ガベージ・コレクタによって削除されます。

テンプレート：テンプレートは、文字列リテラルの新たな形態で、バッククォートが区切り文字になっています。テンプレートを使うと、式の埋込みや、複数行にわたる文字列の記述を行うことができます。Nashornは、関数を使って出力する、ES6の「タグ付き」文字列もサポートしています。

```
// 複数行の文字列
'string text line 1
 string text line 2'
// 式の埋込み
'string text ${expression} string text'
// 「tag」関数を使って出力
tag 'string text ${expression} string text'
```

なお、–scriptingモードでは、従来どおり$EXEC式の簡略表記としてバッククォートが使われることに注意してください。

バイナリと8進数値：さらに、バイナリと8進の数値リテラルに新しい構文が追加されています。

```
// バイナリ数値リテラル
0b111110111 === 503
// 8進数値リテラル
0o767 === 503
```

このようなES6の機能に加え、Nashornの利便性を向上させる新機能も追加されています。

JavaScript Parsing API

JDK 9では、NashornのParser APIが拡張され、ES6の構文が完全にサポートされています。次に示すのは、サンプルのJavaScriptを解析する例です。

```
// Nashornリソースからparser.jsをロード
//// パーサー・ライブラリをロード
// ES6のクラス宣言を含むサンプル・スクリプト
var script = "class XYZ {}";

// スクリプトを解析し、抽象構文木を構築
var json = parse(script);

// 抽象構文木をJSONに変換して出力
print(JSON.stringify(json, undefined, 4));
```

この結果、次のJSONが出力されます。

```
{
  "type":"Program",
  "body":[
    {
      "type":"VariableDeclaration",
      "declarations":[
        {
          "type":"VariableDeclarator",
          "id":{
            "type":"Identifier",
            "name":"XYZ"
          },
          "init":{

          }
        }
      ]
    }
  ]
}
```

Parsing APIは、コードの分析や計測機能のインジェクションにも使用できるため、開発者は高い柔軟性を実現できます。NetBeansは、JavaScript機能の処理のほとんどでこのAPIを使用しています。

$EXEC 2.0

Nashornで特に評判のよい機能は、シンプルなバッククォート式を使ってプロセスをフォークできることです。珍しいこの機能によって、開発者はシェル・スクリプトをJavaScriptで書くことができます。

```
var listing = `ls –l`;
```

この例では、ls –lコマンドから出力される文字列がlistingに格納されます。

バッククォート式は、$EXEC関数の簡略表記です。つまり、`ls –l`は $EXEC('ls -s')を簡潔に表したものです。

JDK 9の$EXECは、以前のバージョンから大きく改善されています。最初の引数を文字列配列で渡すことができるようになったため、空白などの特殊文字を含む引数を渡せるようになりました。

```
$EXEC(["/bin/echo", "my.java", "your java"]);
```

$EXEC関数には、throwOnErrorプロパティも追加されています。これをtrueに設定すると、コマンドが失敗した際にRangeErrorが発生します。今までと同じように、$EXITグローバルを使ってコマンドのステータスを確認することも可能です。

```
$EXEC.throwOnError = true
`javac my.java`
<shell>:1 RangeError:
   $EXEC returned non-zero exit code:2
```

[編集注：最後の2行は1行で表示されます。]管理性も向上しており、コマンドの引数として標準のリダイレクト・ディレクティブを指定し、コマンドのI/O操作を行うこともできます。

```
$EXEC("echo 'my argument has spaces' > tmp.txt");
```

$EXECの最後の3つの引数に入力ストリームと出力ストリームを渡し、stdin、stdout、stderrをオーバーライドすることもできます。

また、セミコロンや改行を使って、複数のコマンドを実行できます。

```
$EXEC(<<EOD);
echo this ; echo that
echo whatever you want
EOD
```

縦棒記号を使って、コマンドの結果を次のコマンドにパイプすることができます。

```
$EXEC("echo 'my argument has spaces' | cat");
```

さらに、組込みコマンドである疑似cd、setenv、unsetenvを使って、環境変数の値を即座に変更することもできます。

```
$EXEC("setenv PATH ~/bin:${ENV.PATH}; mycmd");
$EXEC("cd ~/bin; ls -l");
```

REPLとしてのjjs

JDK 8版のNashornでは、新しいコマンドライン・ツールJava JavaScript (jjs) が追加されました。jjsにより、開発者はJavaコードを書かずにNashorn機能のテストやJavaScriptアプリケーションの起動を簡単に行えるようになります。

JDK 9では、jjsの入出力APIが新しいコンソール入力処理ライブラリjline2に置き換えられています。この変更によって、開発者は、シェルの使用時に、想定されたすべての標準制御を利用できるようになります。次に例を示します。

- 左右の矢印で入力を移動、オプション・キーを押しながら矢印を押すとシンボル単位でジャンプ
- 前方削除および後方削除機能
- [Ctrl]キーを押しながら[k]キーを押すと行の以降の部分を削除、[Ctrl]キーを押しながら[y]キーを押すとその行を復元
- 上下矢印で入力履歴をスクロール表示
- タブ補完をグローバル、プロパティ、Java型にも拡張
- VT100エスケープ・シーケンスによる画面フォーマット

こういった細かい変更すべてによって、作業が少し楽にできるようになっ

ています。

パフォーマンス

JDK 9では、デフォルトで楽観的型最適化がオンになっています。つまり、Nashornでは、エンジンが強い型付けを決定するにつれて、時間とともにコードのパフォーマンスが向上します。

```
function f(array, i) {
  return array[i] + array[i + 1];
}
```

上記の例では、Nashornはまずiとarrayの中身が整数であると仮定（プラス記号があるためです）して関数fの最適化を行います。この仮定が正しくないことがわかった場合、Nashornは実際に使用されている型で新しいバージョンの関数を作成し、それを使って古いコードをホットスワップします。この最適化は呼び出し側ごとに特化して行われるため、常に最高のソリューションが使用されます。

まとめ

本記事では、JDK 9のNashornエンジンが提供する多くの便利な新機能について見てきました。ここで紹介したのは今回のリリースでの新機能ですが、パイプラインには他にも手が加えられている点が多くあります。そういった機能もまもなくお披露目できると思います。</article>

Jim Laskey（@wickund）：オラクルのJavaプラットフォーム・グループで上級開発マネージャーを務める。1970年代中盤からコンパイラ・ランタイムの開発を行っており、Symantec、Apple、Azul Systemsで働いた経験を持つ。

LONDON JUG

London Java Community（LJC）は、LJC JUG（@ljcjug）としても知られています。LJCには、ヨーロッパや国際的に活躍する開発者が参加しています。LJCは、ロンドンのJava関連人材紹介会社RecWorksのBarry Cranford氏により、知識やスキルを共有したいという同じ目的を持った開発者が集まる場所として2006年に設立されました。それ以来、LJCは発展を続け、現在の会員は6,000名を超えています。幸運なことに、LJCは早い時期から、Zoe Slattery氏、Ben Evans氏、Martijn Verburg氏、Simon Maple氏、John Stevenson氏、Trisha Gee氏といった多くのベテランJava開発者に率いられることになりました。

LJCはJava Community Process（JCP）にも積極的に関与しており、第11回Duke's Choice Awardを受賞したほか、JCP Member of the Yearも共同受賞しています。

LJCは、RecWorksのスタッフによるサポートも受けつつ、毎月3回から4回のイベントを開催しています。その1つが、定期的に開催されているHackTheTowerというハッカソン・イベントです。参加者がグループを組んでOpenJDKを使ったプログラムを書き、ScalaやClojureのプロジェクトを作成し、独自に長年温めてきた計画に取り組んでいます。また、LJCは、年次のオープン・カンファレンス（UnConf）を開催するほか、Devoxx UKとDevoxx4Kidsの支援も毎年行っています。こういったカンファレンスは、ブリストル、マンチェスターなどにあるイギリスの他のJUGとも密接に協力しながら開催されています。

さらに、LJCメンバーは自家製のライブラリやフレームワークを共有し、メーリング・リストで質問に答え、Adopt-a-JSRやAdopt OpenJDKなどの、JUGが計画したプロジェクトに参加しています。興味がある方はぜひご参加ください。