Portable Database Access for JavaScript Applications using Java 8 Nashorn

Kuassi Mensah
Director, Product Management
Server Technologies
October 04, 2017
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Speaker Bio

• Director of Product Management at Oracle
  (i) Java integration with the Oracle database (JDBC, UCP, Java in the database)
  (ii) Oracle_datasource for Hadoop (OD4H), upcoming OD for Spark, OD for Flink and so on
  (iii) JavaScript/Nashorn integration with the Oracle database (DB access, JS stored proc, fluent JS )

• MS CS from the Programming Institute of University of Parıs

• Frequent speaker
  JavaOne, Oracle Open World, Data Summit, Node Summit, Oracle User groups (UKOUG, DOAG, OUGN, BGOUG, OUGF, GUOB, ArOUG, ORAMEX, Sangam, OTNYathra, China, Thailand, etc),

• Author: Oracle Database Programming using Java and Web Services

• @kmensah, http://db360.blogspot.com/, https://www.linkedin.com/in/kmensah
Program Agenda

1. The State of JavaScript
2. The Problem with Database Access
3. JavaScript on the JVM - Nashorn
4. Portable Database Access
5. Wrap Up
Program Agenda

1. The State of JavaScript
2. The Problem with Database Access
3. JavaScript on the JVM - Nashorn
4. Portable Database Access
5. Wrap Up
The State of JavaScript
JavaScript & Java: Top Two Popular Languages
The State of JavaScript – The Good

After 22 years since inception, JavaScript has become an ecosystem

• Douglas Crockford: JavaScript the Good Parts & New Good Parts in ES6
  “a beautiful, elegant, lightweight and highly expressive language” http://bit.ly/2xaxX9Q

• Tons of libraries, frameworks, and tools

• Versatile: all programming models, styles and applications types

• JavaScript extended thru other Languages @ http://bit.ly/1VpgaOl

• Ubiquitous: browser, middle-tier, database
The State of JavaScript – The Bad and the Ugly

• “the JavaScript ecosystem is both thriving and confusing as all hell”

   A Brief, Incomplete History of JavaScript @ http://bit.ly/2haZOAb

• Many flavors
  – ECMAScript, TypeScript, CoffeScript, PureScript, etc.

• The JavaScript fatigue
  – Overwhelming number of frameworks and libraries: jQuery, Node.js, Backbone.js, Angular.js, Require.js, Grunt.js, React.js, Redux, Webpack, Babel, Gulp, ....
    • Frameworks wars: e.g., Node.js vs io.js
  – The stress of being behind

• The problem with database access
Program Agenda

1. The State of JavaScript
2. The Problem with Database Access
3. JavaScript on the JVM - Nashorn
4. Portable Database Access
5. Wrap Up
The Problem with Database Access
Client-side JavaScript (browser) should not directly access database.

Server-side JavaScript database access

- Proprietary database APIs
  - StrongLoop for node-oracledb
  - ActiveXObject(“ADODB.Connection”) for SQL Server, Access,

- Node.js cross-db ORMs
  - Sequelize: for Node.js and io.js; supports PostgreSQL, MySQL, MariaDB, SQLite and MS SQL
  - Bookshelf.js (Node.js); supports PostgreSQL, MySQL, MariaDB, and SQLite
  - CaminteJS (Node.js): adapters for mysql, sqlite3, riak, postgres, couchdb, mongodb, redis, neo4j, firebird, rethinkdb, tingodb

- Missing a standard database access library
Program Agenda

1. The State of JavaScript
2. The Problem with Database Access
3. JavaScript on the JVM - Nashorn
4. Portable Database Access
5. Wrap Up
JavaScript on the JVM - Nashorn
JavaScript on the JVM - Nashorn

- Project Nashorn: replacement for Rhino
  https://blogs.oracle.com/nashorn/
  https://wiki.openjdk.java.net/display/Nashorn/Nashorn+jsr223+engine+notes
  https://docs.oracle.com/javase/8/docs/technotes/guides/scripting/nashorn

- Built into Java 8

- Interoperability bw Java & JavaScript
  - Use Java libraries in JavaScript i.e., JDBC (standard Java API), jOOQ (open source)

- New features in Java 9
  - JEP 236: Parser API for Nashorn
  - JEP 292: Implement Selected ECMAScript 6 Features in Nashorn
Invoking JavaScript Procedures/Functions with Nashorn

```
jjs
```

```
java
```

```
javax.script
wrapper
```

```
Function/Procedure
```

```
JS
```

```
Function/Procedure
```

```
JS
```
Running JavaScript with the Nashorn Command-Line

$JAVA_HOME/bin/jjs
jjs> var hw = "Hello World"
jjs> hw
Hello World
jjs> exit()

hello.js
#!/usr/bin/jdk8/bin/jjs
print("Hello from JavaScript");
Hello from JavaScript

hello2.js
var hellow = "Hello from JavaScript!";
return hellow;
}
var output = hello();
print(output);
jjs hello.js
Hello from JavaScript!
Running JavaScript with Nashorn using *javax.script*

1. Create an instance of the script engine manager
   ```java
   ScriptEngineManager factory = new ScriptEngineManager();
   ```

2. Create an instance of a Nashorn engine
   ```java
   ScriptEngine engine = factory.getEngineByName("javascript");
   ```

3. Evaluate the JavaScript code
   ```java
   engine.eval("load('foo/bar/myScript.js')");
   ```

4. Invoke the JavaScript function using `invokeFunction`
   ```java
   engine.invokeFunction("myFunction");
   ```
Running JavaScript with Nashorn using `javax.script`

1. Create an instance of the script engine manager
   ```java
   ScriptEngineManager factory = new ScriptEngineManager();
   ```

2. Create an instance of a Nashorn engine
   ```java
   ScriptEngine engine = factory.getEngineByName("javascript");
   ```

3. Initialize your Java resource
   ```java
   URL url = Thread.currentThread().getContextClassLoader().getResource(abFilePath);
   ```

4. Evaluate the JavaScript code
   ```java
   engine.eval(new InputStreamReader(url.openStream()));
   ```

5. Invoke the evaluated JavaScript function using `invokeFunction`
   ```java
   Invocable invobj = (Invocable) engine;
   Object selectResult = invobj.invokeFunction("selectQuery", inputId);
   ```
Program Agenda

1. The State of JavaScript
2. The Problem with Database Access
3. JavaScript on the JVM - Nashorn
4. Portable Database Access
5. Wrap Up
Portable Database Access
Portable JavaScript Database Access with JDBC

In JavaScript (with Nashorn)
1. Set the vendor driver
2. Set the connect string
3. Get a JDBC connection
4. Create and run JDBC Statements or PrepStmt
5. Get ResultSets
6. Close Rset, Stmts & Conn
JavaScript *getConnection()* – MySQL database

```javascript
var getConnection = function () {
    var Properties = Java.type("java.util.Properties");
    var System = Java.type("java.lang.System");
    var Driver = Java.type("com.mysql.jdbc.Driver");
    var DM = Java.type("java.sql.DriverManager");
    var oracleDriver = new Driver();
    var conn = null;
    var url = null;
    url = "jdbc:mysql://localhost:3306/mydb";
    var props = new Properties();
    props.setProperty("user", "hr");
    props.setProperty("password", "hr");
    conn = DM.getConnection(url, props);

    return conn;
}
```
JavaScript `getConnection()` – Oracle database

```javascript
var getConnection = function () {
    var Properties = Java.type("java.util.Properties");
    var System = Java.type("java.lang.System");
    var Driver = Java.type("oracle.jdbc.OracleDriver");
    var DM = Java.type("java.sql.DriverManager");
    var oracleDriver = new Driver();
    var conn = null;
    var url = null;
    url = "jdbc:oracle:thin:@//localhost:1521/JdbcNashorn";
    var props = new Properties();
    props.setProperty("user", "hr");
    props.setProperty("password", "hr");
    conn = DM.getConnection(url, props);
    return conn;
}
```
queryEmp using JDBC

function queryEmp()
{
    var conn1 = getConnection();
    var output = "";
    var Stmt = conn1.createStatement();
    var output = "ID   First         Last
         
";
    var RSet = Stmt.executeQuery("SELECT employee_id, first_name, last_name from employees where rownum < 10");
    while (RSet.next()) {
        output = output + RSet.getString(1) + " " + RSet.getString(2) + " " + RSet.getString(3) + "\n";
    }
    RSet.close(); Stmt.close(); conn1.close();
    return output;
}

var foo = queryEmp();
print(foo);
Calling \textit{queryEmp} with JJS

\texttt{jjs -cp ojdbc8.jar:. queryEmp.js}

<table>
<thead>
<tr>
<th>ID</th>
<th>First</th>
<th>Last</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Steven</td>
<td>King</td>
</tr>
<tr>
<td>101</td>
<td>Neena</td>
<td>Kochhar</td>
</tr>
<tr>
<td>102</td>
<td>Lex</td>
<td>De Haan</td>
</tr>
<tr>
<td>103</td>
<td>Alexander</td>
<td>Hunold</td>
</tr>
<tr>
<td>104</td>
<td>Bruce</td>
<td>Ernst</td>
</tr>
<tr>
<td>105</td>
<td>David</td>
<td>Austin</td>
</tr>
<tr>
<td>106</td>
<td>Valli</td>
<td>Pataballa</td>
</tr>
<tr>
<td>107</td>
<td>Diana</td>
<td>Lorentz</td>
</tr>
<tr>
<td>108</td>
<td>Nancy</td>
<td>Greenberg</td>
</tr>
</tbody>
</table>
**selectQuery Function**

```javascript
var selectQuery = function(id)
{
    var prepStmt;
    var output = "";
    var conn1 = getConnection();
    if(id == 'all') {
        prepStmt = conn1.prepareStatement("SELECT a.data FROM jemployees a");
    } else {
        prepStmt = conn1.prepareStatement("SELECT a.data FROM jemployees a WHERE a.data.EmpId = ?");
        prepStmt.setInt(1, id);
    }
    // execute Query
    var resultSet = prepStmt.executeQuery();
    while(resultSet.next()) {
        output = output + resultSet.getString(1) + "<br>";
    }
    resultSet.close(); prepStmt.close(); conn1.close(); return output;
}
```
public class InvokeScript2 {
    public static void main (String[] args) {
        String output = new String();
        try {
            // create a script engine manager
            ScriptEngineManager factory = new ScriptEngineManager();
            // create a JavaScript engine
            ScriptEngine engine = factory.getEngineByName("javascript");
            // read the script as a java resource
            engine.eval(new InputStreamReader(InvokeScript2.class.getResourceAsStream("select2.js")));
            Invocable invocable = (Invocable) engine;
            Object selectResult =
                invocable.invokeFunction("selectQuery", args[0]);
            output = selectResult.toString();
        } catch(Exception e) {
            ByteArrayOutputStream baos = new ByteArrayOutputStream(4096);
            e.printStackTrace(new PrintStream(baos));
            output = e + baos.toString() + e.getMessage();
        }
        System.out.println(output);
Calling `selectQuery` with `javax.script`

```java
java -cp ojdbc8.jar:. InvokeScript2 "100"
```

```java
java -cp ojdbc8.jar:. InvokeScript2 "$1"
{
    "EmpId" : "100",
    "FirstName" : "Kuassi",
    "LastName" : "Mensah",
    "Job" : "Manager",
    "Email" : "kuassi@oracle.com",
    "Address" : {
        "City" : "Redwood",
        "Country" : "US"
    }
}
```

```text
ade:[ kmensah_demo ] [kmensah@adc00pef ~/CosmosDB]$ 
```
JavaScript In the Database

Rationales

• Many DBMS(es) furnish JavaScript
  - MongoDB
  - DocumentDB/CosmosDB
  - ForerunnerDB

• The JVM in Oracle Database (OJVM) Supports Java 8 Nashorn

• Reuse JavaScript code and skills for database modules

• Avoid data shipping
  i.e., processing thousands/millions of JSON documents in RDBMS
JavaScript Database Access with DB Resident JDBC

• 1 call
• Low network latency in WAN or Cloud environments
• Can be invoked by SQL or any database client
The Embedded Java VM in Oracle Database

Database Process / Session (PGA)

Newspace
Oldspace
Sessionspace

Shared Memory (SGA)

JDK8 VM (+ Nashorn) Class Objmems
Internspace

User Java & JavaScript Classes
Portable JDBC Connect String

Setting the JDBC URL depending on where the code is running

```java
if (System.getProperty("oracle.jserver.version") !== null)
    // your code is running within the database JVM
    {
        url = "jdbc:default:connection:";
        conn = oracleDriver.defaultConnection();
    }
else
    // your code is running in a client JVM (outside the database)
    {
        url = "jdbc:oracle:thin:@//localhost:1521/pdb1.localdomain";
        var props = new Properties();
        props.setProperty("user", "hr");
        props.setProperty("password", "hr");
        conn = DM.getConnection(url, props);
```
How to Run JavaScript In Oracle Database

1. Load your JavaScript code in your Schema as Java Resource
   
   `loadjava -v -u yourschema myJS.js`

2. Invoke JavaScript: 3 approaches
   1) Using `dbms_javascript.run()` from SQL or PL/SQL
      
      SQL: `call dbms_javascript.run('myJS.js');`
      PL/SQL: `dbms_javascript.run('myJS.js');`

   2) Using `DbmsJavaScript.run()` from Java in the database
      
      `import oracle.aurora.rdbms.DbmsJavaScript;`
      `DbmsJavaScript.run(“myJS.js”);`

   3) using the standard JSR223 scripting APIs `javax.script`
Running `queryEmp.js` in the Database

3 Easy steps

1. Load `queryEmp.js` in the user schema

   ```bash
   loadjava -r -v -u hr/hr queryEmp.js
   ```

2. `runEmp.sql`: a SQL script for ease of invocation

   ```sql
   set serveroutput on
   call dbms_java.set_output(5000);
   call dbms_javascript.run('queryEmp.js');
   ```

3. Shell script for invoking the SQL script

   ```bash
  #!/bin/bash
   set -v
   sqlplus hr/hr @runEmp.sql
   ```
Running *selectQuery* in the Database

3 Easy steps

1. Load InvokeScript2.java in the user schema
   
   ```
   loadjava -r -v -u hr/hr InvokeScript2.java
   ```

2. Create a SQL wrapper for invoking invokeScript2.main()
   
   ```
   CREATE OR REPLACE procedure invokeScript(inputId varchar2) as language java
   name 'InvokeScript2.main(java.lang.String[])';
   ```

3. Shell script for invoke the SQL wrapper
   
   ```
   set serveroutput on
   call dbms_java.set_output(5000);
   Call invokescript('&1');
   ```
Program Agenda

1. The State of JavaScript
2. The Problem with Database Access
3. JavaScript on the JVM - Nashorn
4. Portable Database Access
5. Wrap Up
Wrap up
Wrap Up

- Java 8 Nashorn allows interoperability between Java and JavaScript
- JavaScript running on the JDK/JRE can leverage the standard JDBC API to ensure portable database access across RDBMSes
- The same JavaScript code may also run within the database using the embedded JVM

Quod Erat Demonstrandum!

All the code samples will be posted on GitHub @ http://bit.ly/2yfZXcZ
Integrated Cloud
Applications & Platform Services