ORACLE®

Multi-language JDI? You're Joking, Right?

Jim Laskey
Multi-language Lead
Java Language and Tools Group

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.

Why?

- Talk server as a discussion point for MLVM Language debugging.
- Print statements are a pain.
- Debugging infrastructure is already there.
  - JPDA works for Java.
  - IDEs already understand JPDA well.
- Server side remote debugging (Node.jar)

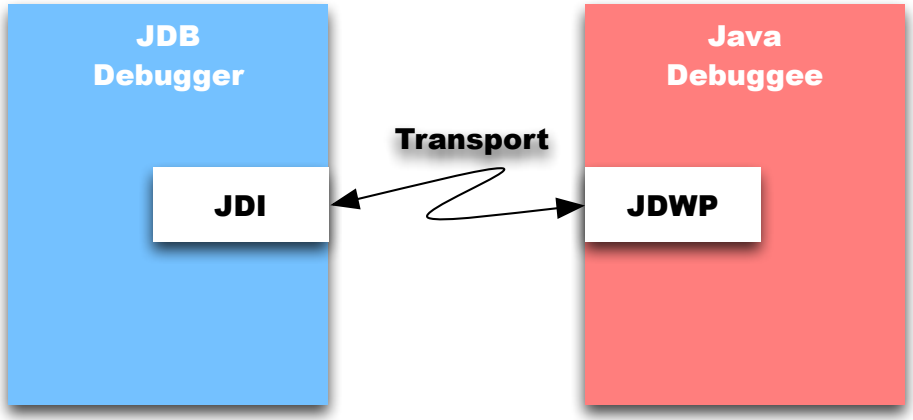
Agenda

- JPDA, JDWP and JDI
- Framework For Multi-Language Debugging
- Specific Examples In Nashorn
- Demo
- Multi-Language JDI?
- Q & A

JPDA, JDWP and JDI

JPDA, JDWP and JDI

- JPDA - Java Platform Debugger Architecture
- JDWP - Java Debug Wire Protocol
- JDI - Java Debugger Interface
  - http://docs.oracle.com/javase/7/docs/jdk/api/jpda/jdi/index.html



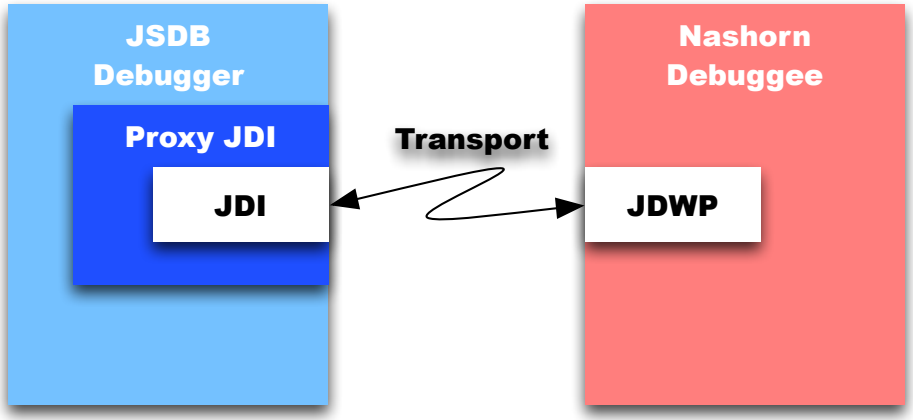
What does JDI do?

- Manage the connection to the debuggee.
  - Connection
    - Sockets
    - Shared memory (Windows)
    - Launch/Attachment
- Sends requests to debuggee.
  - Setting breakpoints
  - Stepping
  - View stacks, frames, objects
- Receives events from debuggee.
  - Class loaded
  - Breakpoint hit
  - Value of objects

Framework For Multi-Language Debugging

Virtualized View Our World Via A Proxy JDI

- Have the proxy connect to Java JDI and transform requests and events.
- Wrap JDI objects in mirroring proxy objects.



VirtualMachine Object

- Abstraction of the (remote) VM.
  - Handles “reflection”-like requests to the debuggee VM.
- Object factory.
  - interns objects.

```
public class NashornJDIVirtualMachine extends NashornJDIMirror
                                       implements VirtualMachine {
```

Mirroring of JDI objects

```
public class NashornJDIMirror implements Mirror {
    private final NashornJDIVirtualMachine nashornvm;
    private final Mirror underlying;

    public NashornJDIMirror(NashornJDIVirtualMachine nashornvm,
                            Mirror underlying)...

    protected NashornJDIVirtualMachine virtualMachine() ...
    protected Mirror underlying()...
}
public class NashornJDIObject extends NashornJDIMirror
                              implements JDIObject {
  ...
  public Object request(Object args...) {
      return underlying().request(virtualMachine(), args);
  }
}
```

Example StackFrame

```
public class NashornJDIStackFrame extends NashornJDIMirror
                                   implements StackFrame {

    public NashornJDIStackFrame(NashornJDIVirtualMachine nashornvm,
                                StackFrame underlying) {
        super(nashornvm, underlying);
    }
...
    public void setValue(LocalVariable local, Value value)
                throws InvalidTypeException, ClassNotLoadedException {
        underlying().setValue(NashornJDIWrapper.unwrap(local),
                              NashornJDIWrapper.unwrap(value));
    }
...
}
```

JDI Object Wrapping/Unwrapping

```
public class NashornJDIWrapper {
...
    public static NashornJDIField wrap(
                NashornJDIVirtualMachine nashornvm,
                Field field) {
        return (field == null)? null : nashornvm.field(field);
    }
...
}
public class NashornJDIVirtualMachine extends NashornJDIMirror
implements VirtualMachine {
...
    protected NashornJDIField field(Field field) {
        return new NashornJDIField(this, field);
    }
...
}
```

Specific Examples In Nashorn

Nashorn Needs

- Debug JavaScript
  - Make JavaScript objects look like classes
  - Make properties look like Java fields
  - Map source code (not always from a file)
  - Make functions look like Java methods
  - Make internal types look like Java values
  - Ignore runtime support when stepping and stack crawls
- Flavours (consideration)
  - JavaScript only
  - JavaScript into Java code
  - Java with JavaScript objects

Make Nashorn objects look like Java objects

```
public static NashornJDIObjectReference
    wrap(NashornJDIVirtualMachine nashornvm, ObjectReference ref) {
        ...
        if (nashornvm.isScriptObjectType(classType)) {
            return new NashornJDINashornObjectReference(nashornvm,
                                                        ref);
        ...
 }
```

...Make Nashorn objects look like Java objects

```
public boolean isScriptObjectType(NashornJDIClassType subClass) {
    return subclassOf(subClass, nashornScriptObjectType());
}
... //NashornJDIVirtualMachine
public static final String NASHORNJDI_SCRIPTOBJECT_TYPE_NAME =
                           "com.oracle.nashorn.runtime.ScriptObject";
private NashornJDIClassType nashornScriptObjectType;
public synchronized NashornJDIClassType nashornScriptObjectType() {
    if (nashornScriptObjectType == null) {
        List<ReferenceType> refTypes =
                    classesByName(NASHORNJDI_SCRIPTOBJECT_TYPE_NAME);
        nashornScriptObjectType = refTypes.isEmpty() ? null :
                                  (NashornJDIClassType) refTypes.get(0);
    }
    return nashornScriptObjectType;
}
```

Make Nashorn properties look like Java fields

```
    } else if (nashornvm.isPropertyMapType(classType)) {
        return new NashornJDINashornPropertyMap(nashornvm, ref);
    } else if (nashornvm.isPropertyType(classType)) {
        return new NashornJDINashornProperty(nashornvm, ref);
...

public class NashornJDINashornObjectReference extends
             NashornJDIObjectReference implements ObjectReference {
    ...
    public NashornJDINashornPropertyMap getMap() {
        return (NashornJDINashornPropertyMap)getFieldValue("map");
    }
    ...
    public List<NashornJDINashornProperty> getMapProperties() {
        return getMap().getProperties();
    }
```

...Make Nashorn properties look like Java fields

```
public class NashornJDINashornObjectReference extends
...
    public NashornJDIValue get(NashornJDINashornProperty property) {
        return invokeMethod("get", property.invokeMethod("getKey"));
    }
...
    public void set(NashornJDINashornProperty property,
                    NashornJDIValue value) {
        invokeMethod("set", property.invokeMethod("getKey"),
                             value);
    }
...
}
```

Map source code

- Source can come from an url or an eval.
- Can’t just read from class name or source
- Need to invoke a request to runtime to get the source.

```
String filename = loc.sourceName();
if (filename.endsWith(".js") || filename.endsWith("<eval>")) {
    ReferenceType scriptType = loc.declaringType();
    Field sourceField = scriptType.fieldByName("source");
    ObjectReference source =
                  (ObjectReference)scriptType.getValue(sourceField);
    ClassType sourceType = (ClassType)source.type();
    Method getStringMethod =
                   sourceType.methodsByName("getString").get(0);
    StringReference sourceStringReference =
    (StringReference)source.invokeMethod(currentThread.getThread(),
                          getStringMethod, new ArrayList<Value>(),
                          ClassType.INVOKE_SINGLE_THREADED);
    String sourceString = sourceStringReference.value();
```

Make Nashorn functions look like Java methods

- Nothing required since source maps through line number tables.
  - Nashorn's tokens track source position.
- Frame values can be access thru LocalVariables.
  - Another form of property in Nashorn.

```
public NashornJDIValue getValue(LocalVariable local) {
    if (local instanceof NashornJDINashornLocalProperty) {
        NashornJDINashornLocalProperty localProperty =
                                (NashornJDINashornLocalProperty)local;
        return localProperty.getScope().get(localProperty);
    }
    return NashornJDIWrapper.wrap(virtualMachine(),
            underlying().getValue(NashornJDIWrapper.unwrap(local)));
}
```

Ignore runtime support

```
public class NashornJDIEventRequestManager extends NashornJDIMirror
                                        implements EventRequestManager {
...
// same for createMethodExitRequest and createStepRequest
public NashornJDIMethodEntryRequest createMethodEntryRequest() {
    NashornJDIMethodEntryRequest request =
                        NashornJDIEventRequest.wrap(virtualMachine(),
                            underlying().createMethodEntryRequest());
    if (virtualMachine().nashornCodeOnly) {
        request.addClassFilter("com.oracle.nashorn.scripts.*");
    }
    return request;
}
```

Ignore runtime support

```
request.addClassExclusionFilter("com.oracle.nashorn.api.*");
request.addClassExclusionFilter("com.oracle.nashorn.codegen.*");
request.addClassExclusionFilter("com.oracle.nashorn.ir.*");
request.addClassExclusionFilter("com.oracle.nashorn.objects.*");
request.addClassExclusionFilter("com.oracle.nashorn.parser.*");
request.addClassExclusionFilter("com.oracle.nashorn.runtime.*");
request.addClassExclusionFilter("com.oracle.nashorn.tools.*");
```

DEMO

Multi-Language JDI?

Multi-Language JDI?

- Is there a demand for it?
- Boiler plate
- MOP(Dynalink)-like based JDWP?

- Debugging MethodHandles/Lambdas?

Boiler Plate

- Providing a stripped down boiler plate.
  - s/xyzzyx/myStuff/
  - s/Xyzzyx/MyStuff/
  - mv XyzzyxJDI??? MyStuffJDI???

http://wiki.jvmlangsummit.com/Multi-language_JDI%3F_You%27re_Joking,_Right%3F

Hooking Up An IDE

- Adding a layer on top of your version of the JDI.
- http://bits.netbeans.org/dev/javadoc/org-netbeans-api-debugger/index.html
- http://confluence.jetbrains.net/display/IDEADEV/PluginDevelopment

Contact

Jim Laskey

james.laskey@oracle.com

Q & A

The preceding 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.

Q&A