Inline::Java

beyond the POD

William Cox (cpan: MYDMNSN)
David Larochelle (cpan: DRLARO)

Outline

Introduction
Why Java? [David]
Basic Instantiation [Will]
Default [David] vs. Shared JVM mode vs. JNI [Will]
Installation [Will]
Case Studies
    Conditional Random Fields with Mallet [David]
    Using Maven [Will]
Tips and Tricks [Will and David]
Other Inline::Java Users
Questions

About William

- Perl developer for 7 years
- $work for Black Duck Software
- Main focus recently has been stream data processing
- I will literally try to make *anything* work with Perl

About David

- Longtime Perl developer
- $work for Berkman Center for Internet & Society
- $project is mediacloud.org
  - Machine Learning
  - Natural Language Processing
  - Big Data
  - Keeping the system running

Why Inline::Java?

- Lots of useful logic is locked up inside languages like Java
- Ultimate in code re-use
- Combine well established projects from either language

Why Java? (Technical Reasons)

- Performance
  - Close to C/C++ speeds & sometimes faster
  - Much faster than Perl/Python/Ruby/etc.
- Scales well in large projects
- Platform Independant
- Object Oriented
- Robustness
  - Forced Error Checking
  - Type safety
- Concurrency

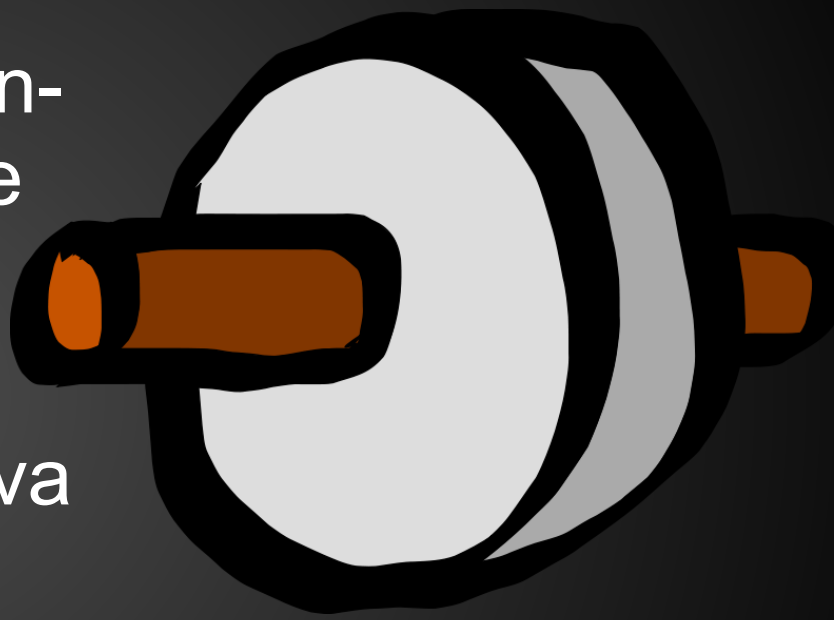
Why Java? (Social/ Nontechnical Reasons and Networks Effects)

- Large community
- Lots of Existing Libraries
  - Maven Central larger than CPAN
- Most Apache Foundation Projects are Java
- Networks effects matter but there are domains where interpreted languages don't make sense:
  - HPC, Big Data

Apache Open Source Libraries

Why not just reimplement in Perl?

Many Java Libraries are non-trivial -- often why they were written in Java



Compatibility w/ existing Java code may be necessary

Don't reinvent the wheel

Example Plucene

Port of Lucene to Pure Perl

Last Release in 2006

Now abandoned

Lesson: Don't Reinvent the Wheel

Installation

- It's 4am, do you know where your `J2SDK/JAVA_HOME` is?
  - Defaults for OS X don't really work, need to find and set manually: (i.e. `/System/Library/Frameworks/JavaVM.framework`)
- Always build in the optional JNI extension; makes for zoom, zoom.
  - Still must elect to use JNI perl `use Inline 'Java';`
  - Upgrading major versions of Java will likely require a reinstall
- PerlInterpreter option seems buggy (marked experimental)

Basic Incantation

```
use Inline Java => 'EOJ';
    class Greeter {
        public Greeter () { }

        public String greeting (String who) {
            return "Hello, " + who;
        }
    }
EOJ

my $greeter = Greeter->new;
print $greeter->greeting('World') . "\n";
```

- auto serialization of basic types (strings, numbers, arrays, filehandles)
  - caveats with member assignment and multiple method signatures (see coercing)
  - no unblessed hashes, use objects/beans instead
- support for Perl callbacks with Java (not covered here)
- utilities for casting/coercing Perl types to Java types (not covered here)

Operating Modes

- Default mode uses a socket to communicate to separate JVM process
- Shared JVM that can be shared among multiple perl processes
- JNI (Java Native Interface) is available if compiled in (recommended)

Default JVM Mode

- JVM invoked on 1st use Inline::Java
- Each process has a separate JVM
- Inline::Java interfaces with the JVM through the network
- Simple but slower

Shared JVM

- Separate JVM process but allows connections from multiple perl processes
- Can in theory be used across different machines
- Designed with Apache/mod_perl forking in mind
- Caveat: Java objects cannot be shared among Perl processes (i.e. forks)

JNI (Java Native Interface)

- Runs the JVM directly inside the Perl process
- Provides significant speed boost for jobs that need only a single JVM
- Caveat: cannot fork after starting JVM

Case Studies

Conditional Random Fields with Mallet

Problem: Differentiate between article text and ads, navigation, and other cruft

Large Human Generated Training Data

Heuristics worked but had significant errors

Solution: Use a Conditional Random Fields model

CRF & Mallet Continued

- What about CPAN?
- Algorithm::CRF - Perl binding for CRF++
- Not updated since 2007

Grade	Perl version	OS name	OS version
FAIL	5.16.2	GNU/Linux	2.6.32-5-686
UNKNOWN	5.18.0	Windows (Win32)	4.0
UNKNOWN	5.16.0	Windows (Win32)	5.2
FAIL	5.14.4	SunOS/Solaris	2.10
UNKNOWN	5.10.1	Windows (Win32)	5.1
FAIL	5.16.3	GNU/Linux	3.2.0-38-virtual
FAIL	5.14.3	SunOS/Solaris	2.10
FAIL	5.16.1	GNU/Linux	3.2.0-31-virtual
UNKNOWN	5.12.3	Windows (Win32)	5.1
UNKNOWN	5.14.3	Windows (Win32)	4.0
FAIL	5.16.2	Mac OS X	12.2.0
FAIL	5.16.2	GNU/Linux	2.6.32-5-686
FAIL	5.8.8	GNU/Linux	3.2.0-3-amd64
FAIL	5.14.3	GNU/Linux	3.5.6-1-arch
UNKNOWN	5.16.2	Windows (Win32)	4.0
FAIL	5.16.2	NetBSD	6.0

CRF & Mallet

- MALLET -- MAchine Learning for LanguagE
- UMass Amherst
- Open Source Machine Learning Library

Integrating with Mallet

- Stored jars in source control repo
- Determine classpath in a begin statement

```
my $class_path;

BEGIN
{
    my $_dirname      = dirname( __FILE__ );
    my $_dirname_full = File::Spec->rel2abs( $_dirname );
    my $jar_dir = "$_dirname_full/jars";
    my $jars = [ 'mallet-deps.jar', 'mallet.jar' ];
    $class_path = scalar( join ':', ( map { "$jar_dir/$_" } @{ $jars } ) );
}
```

Mallet continued

- Invoke main class instead of calling java through system

```
sub create_model_inline_java
{
...
use Inline (
        Java  => 'STUDY',
        STUDY => [  qw ( cc.mallet.fst.SimpleTagger )   ],
        AUTOSTUDY => 1,  CLASSPATH => $class_path,
        PACKAGE   => 'main'     );
cc::mallet::fst::SimpleTagger->main(  [ "--train", "true", "--iterations", $iterations,
"--model-file", $model_file_name, $training_data_file ] );
... }
```

Mallet continued

- Logic to run the model defined in block of Java included in the Perl file.
- Java code developed in Eclipse and tested externally initially
- Development cycle:
  - Tweak java block in Eclipse
  - Paste into Perl Source
  - Tested
  - Repeat

- Code http://mediacloud.org or https://github.com/berkmancenter/mediacloud

Maven

Problem: Need to gather metadata on every POM in Maven ecosystem.

- Only Maven can truly parse a POM (Project Object Model - XML) file
  - Variable Interpolation
  - Inheritence
- `mvn` provides no utility to output the effective POM

Solution: replicate Java codepath from Perl to introspect the POM

Maven (cont.)

```
my (@class, %class, $maven_home);
BEGIN {
     chomp(my $mvn = `which mvn`);
     $mvn or die "Unable to locate maven installation - is it installed?";
     ($maven_home = Cwd::realpath $mvn) =~ s{/bin/mvn$}{};
     $ENV{CLASSPATH} = join ':', <$maven_home/lib/*.jar>,
          <$maven_home/boot/*.jar>, "$maven_home/conf";
     @class = ('java.io.File', ..., 'org.apache.maven.Maven', ...);
     %class = map { (/\.(\w+)$/ =>
          Inline::Java::java2perl(__PACKAGE__, $_)) } @class;
     my $stash = Package::Stash->new(__PACKAGE__);
     while (my ($cls, $pkg) = each %class) {
          $stash->add_symbol("\&$cls" => sub () { $pkg });
     }
}
use Inline (Java => 'STUDY',
     EXTRA_JAVA_ARGS => "-Dmaven.home=$maven_home
          -Dclassworlds.conf=$maven_home/bin/m2.conf",
     STUDY => \@class, AUTOSTUDY => 1, JNI => 1,
     CLASSPATH => $ENV{CLASSPATH},
     NAME      => __PACKAGE__,
     DIRECTORY => ((__FILE__ =~ m{^(.*)\.pm$})[0] . '/_Inline'),
);
```

Other Interpreted Languages

- Python -- Jython
  - Converts Python to Java Bytecode
  - Seamless ability to call Java classes & Libraries
  - Can't access C modules such as NumPy
- Ruby -- JRuby
- Moe
  - Perl implementation in Scala

Tips and Tricks

- STUDY vs. AUTOSTUDY: auto is useful but limited. Use together for best effect.
- Make shortcut constants to object packages to avoid carpal-tunnel.
- Plan ahead to minimize Perl<->Java calls.

Other Inline::Java Users

- Lingua::StanfordCoreNLP - Perl interface to Stanford NLP
- Net::RabbitMQ::Java - Perl wrapper around Java AMQP library

Questions?

Thank You!

William Cox

mydimension@gmail.com

David Larochelle

drl@Larochelle.name