Exploiting JRuby: Building Domain-Specific Languages for the Java Virtual Machine™

Rob Harrop

VP
Interface21
http://www.interface21.com

TS-9294

Goal of This Talk

What you will gain

Learn how to create and exploit domain-specific languages for the Java Virtual Machine™ (JVM™) using JRuby.

The terms “Java Virtual Machine” and “JVM” mean a Virtual Machine for the Java™ platform.

Agenda

Introduction to DSLs
DSLs in Action
Techniques for DSLs in JRuby
Decomposing DSLs
Elements of DSL Style
Summary
Q&A

Agenda

Introduction to DSLs

DSLs in Action

Techniques for DSLs in JRuby

Decomposing DSLs

Elements of DSL Style

Summary

Q&A

Introduction to DSLs

What is a DSL?

- Domain-specific language
  - Custom language designed for a specific purpose
  - Not just about **business-specific** domains
- General purpose examples
  - Build language (rake)
  - Application management (MScript—coming soon!)
- Business examples
  - Derivative calculation
  - Corporate action entitlement calculation

Introduction to DSLs

Why bother?

- Greatly simplify repetitive tasks
  - Java Management Extensions (JMX™)?
- Encapsulate boilerplate code
- Provide an API that expresses the intent of the code clearly
- Invest a little up front to save a lot in the future

- **DSLs should deliver clear value**

Introduction to DSLs

Use cases for DSLs

- Simplify/encapsulate some API
  - JMX API, for example
- Abstract complex business problems
  - Corporate action processing
- Support operational control of an application
  - Script maintenance tasks

Introduction to DSLs

Classifying DSLs

- Type
  - External—XML
  - Internal—JRuby, Groovy
- Target
  - General purpose—MScript
  - Business-specific—CAScript

Agenda

Introduction to DSLs

DSLs in Action

Techniques for DSLs in JRuby

Decomposing DSLs

Elements of DSL Style

Summary

Q&A

DEMO

DSLs in Action—Rake and MScript

Agenda

Introduction to DSLs

DSLs in Action

Techniques for DSLs in JRuby

Decomposing DSLs

Elements of DSL Style

Summary

Q&A

Techniques for DSLs in JRuby

Operator overloading

- Operator overloading allows natural syntax for common functions
  - Concatenation, addition
  - Attribute/index access

Example from MScript:

```
puts mbean[:SomeAttribute]
mbean[:SomeAttribute] = "Hello World!"
```

Techniques for DSLs in JRuby

Hashes and symbols

- Symbols are a great way to identify objects in your DSL
  - Tasks in rake
- Hashes are great for expressing relationships
  - Task dependency in rake

```
task :run-application => :setup
```

Techniques for DSLs in JRuby

Blocks

- Encapsulate executable logic
  - Paired with symbols to identify this logic
- Task actions in rake

```
task :run-application => :setup do
  ruby "application.rb"
end
```

Techniques for DSLs in JRuby

Method missing

- Great advantage over Java platform
- No need to know all operations in advance—but get natural call syntax
  - JMX technology operations on a MBean proxy in MScript

Techniques for DSLs in JRuby

Dynamic type extension

- Another brilliant advantage over Java platform
- Dynamically add methods to classes and objects
- Attribute access in MScript

Techniques for DSLs in JRuby

Java technology integration

- Manipulate Java platform domain logic using JRuby
- Take full advantage of existing investment
- Add in DSLs where they make sense
- Get the best of both worlds

Techniques for DSLs in JRuby

Options for Java technology integration

- Access Java platform from Ruby using JRuby
  - include_class etc
- Access Ruby from Java platform using JRuby API
  - This is reasonably complex
- Bi-directional access
  - Critical for DSLs that aim to simplify Java technology usage
- Spring integration
  - Create beans using JRuby

DEMO

Techniques for DSLs in JRuby

Agenda

Introduction to DSLs

DSLs in Action

Techniques for DSLs in JRuby

Decomposing DSLs

Elements of DSL Style

Summary

Q&A

Decomposing DSLs

- All DSLs are composed of these common building blocks
- As expected a higher level of abstraction is provided so as not to appear like basic method calls
- Let's dive into the code and look at these techniques in action

DEMO

Decomposing DSLs

Agenda

Introduction to DSLs

DSLs in Action

Techniques for DSLs in JRuby

Decomposing DSLs

Elements of DSL Style

Summary

Q&A

Elements of DSL Style

Identify the problem, express the solution

- DSLs are not simply Java code expressed in a dynamic language
- Identify the problem and create a syntax that clearly expresses the solution

```
task :run_application do
 ruby "application.rb"
end
```

Elements of DSL Style

Metadata and behaviour

- DSL expressions typically contain metadata and behaviour as part of a single expression

```
corporate_action :bonus_issue do |position, payout|
    entitlement :stock =>
      position.quantity * payout.rate
end
```

Don’t let DSLs become another configuration file

Elements of DSL Style

Thinking the Ruby way

- Use type extension in favour of if/else
- Use blocks rather than anonymous types
- Use methods on objects rather than statics

Elements of DSL Style

Characteristics of a DSL

- Limited in scope
  - Address a small part of your domain
- Limited in functionality
  - Only the features needed to address that part

For More Information

- Books
  - *Programming Ruby* (Thomas, Fowler and Hunt)
  - *The Ruby Way* (Fulton)
- Web
  - http://jruby.codehaus.org
  - http://blog.interface21.com

Q&A

Exploiting JRuby: Building Domain-Specific Languages for the Java Virtual Machine™

Rob Harrop

VP
Interface21
http://www.interface21.com

TS-9294