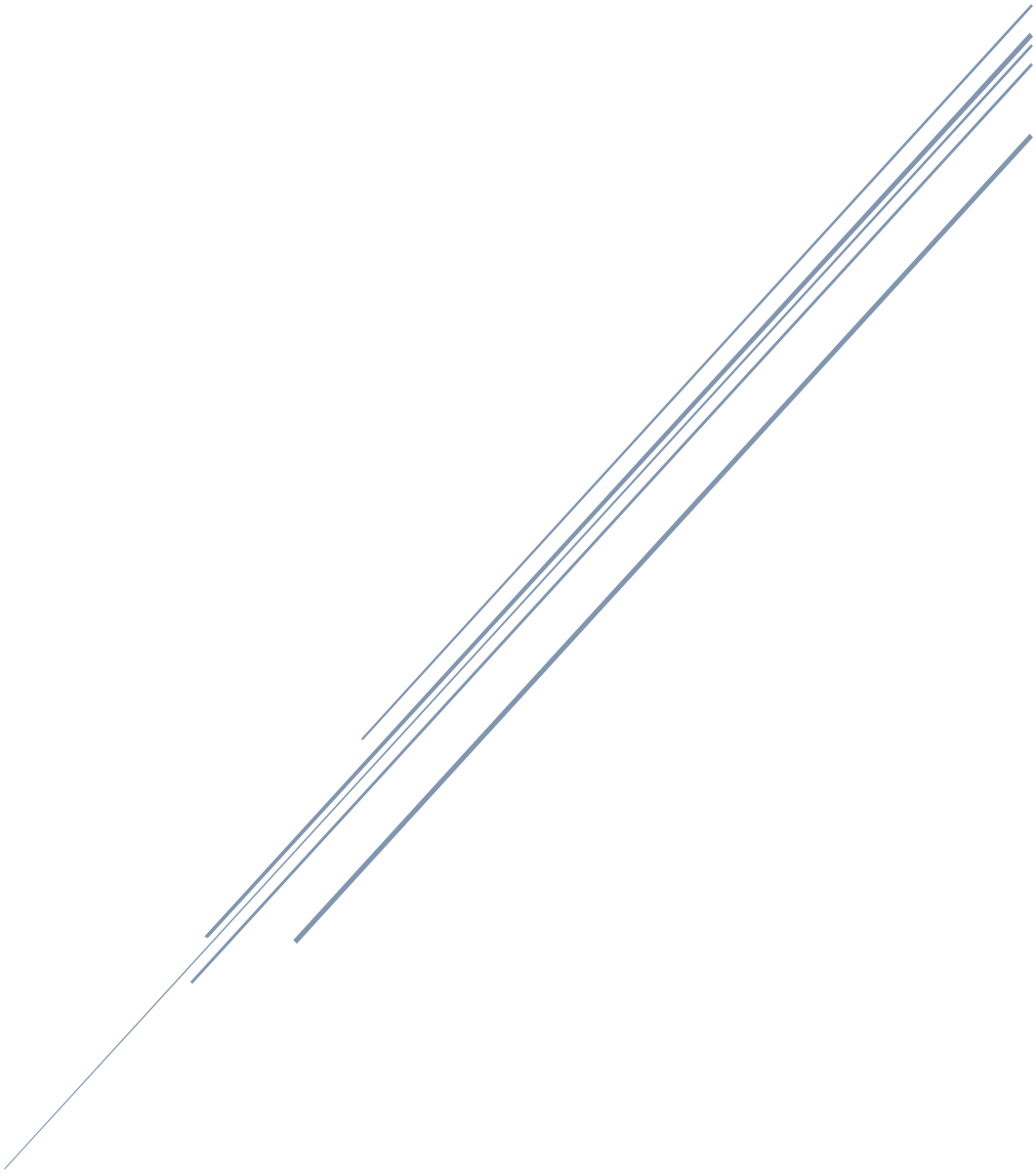
DSL REPORT

KS489



CO7517

Contents

Introduction

There are plenty of computer programming languages used in creating different kinds of Domain-Specific Languages (DSL)s. Scala and JRuby are very popular general purpose programming languages used in designing different kinds of DSLs. The purpose of this document will be to objectively highlight the key differences and similarities between these two languages when creating DSLs and how they address parsing, validation, code generation, integration with libraries, integration with web technologies. There will also be some pros and cons in using these two technologies and what typical scenarios are they used in today's industry.

Language Introduction

Scala and JRuby both use the Java Virtual Machine (JVM) Platform which means both source code will eventually be compiled to bytecode.

Scala

Scala is a multi-paradigm general purpose programming language. Scala is a statically typed language that has object-oriented and functional programming features.

JRuby

JRuby is a flavour of the common general purpose programming language called Ruby which is basically Ruby atop the JVM. JRuby is an object-oriented programming language with dynamic typing.

Parsers

Both languages have support for parsers that are able to transform text-model, model-model and model-text. Below shows the difference between the two languages and how they approach using parsers when creating DSLs. They both have facilities to query models and provide constraints on models using such technologies like QVT (MOF Queries/Views/Transformations) and OCL (Object Constraint Language)

Scala

There are several Scala parsers out there on the web. There is a popular parser on the market called Scala's Parser Combinators that is used very often in making External DSLs.

JRuby

Ruby also has several parsers out there for making internal DSLs.

Below lists the difference between the two languages

	Scala	**JRuby**
DSL Type	Internal, External	Internal
Language Paradigms	Object Oriented, Functional	Object Oriented
Language Type	Statically Typed	Dynamically Typed
JVM Platform	Yes	Yes
Duck Typing	Yes (Statically Checked)	Yes
Runtime Metaprogramming	Yes	Yes
Compile time	Yes	No
Metaprogramming	Yes	Yes

Flexible Syntax	Yes	Yes
Techniques	Operator overloading, Hashes, Higher-order functions, by-name parameter evaluation, implicit definitions and parameters.	Operator overloading, Hashes Symbols, Blocks
Parser to abstract types	Parser Combinator, Metamodels, EMF	Metamodels, EMF

Domain Abstractions and Parsing

These languages do have parsers that will be able to parse your grammar into metamodels or EMF models. Scala does have other ways of abstracting the domain using an object abstraction and the use of functional programming techniques. – "When you design a DSL in Scala, it's mostly an object model that serves as the base abstraction layer. You implement specializations of various model components using subtyping, and form larger abstractions by composing with compatible mixins from the solution domain. For the actions within your model, you create functional abstractions, then compose those using combinators. Figure 6.3 explains the ways you can achieve extensibility in Scala abstractions with the dual power of OO and functional capabilities." – Debasish Ghosh, Jonas Boner. 2010 *DSLs IN ACTION.* Greenwich: Manning

Scala can use a parser combinator for external DSLs. This does mean Scala can expose an internal API that is baked into the language to interact with the abstract objects to be used for pattern matching, validation and code generation. Also this does mean we get interoperability between other DSLs that were created with Scala by using subtyping.

There are various parsers for JRuby and Scala for making internal DSLs. These two languages make use of the Metaprogramming facilities to take advantage of making Internal DSLs such as Reflection.

Validation

Duck Typing

Duck Typing gives type checking to be deferred at runtime compilation which is very useful when making DSLs. Both languages use Duck Typing through either dynamic typing or reflection. Both languages have concise and flexible syntax suitable when trying to write a simple DSL that should focus on the domain rather than the correctives of the programming language using type annotations. This is where dynamically typed languages really shine because type inference is deferred at runtime. Scala's syntax is more complex and it is difficult to understand but most errors can be caught during compile time which makes it terser. Dynamic typed languages such as JRuby avoid this but do require more testing.

Integrations with Libraries

With every good language there needs to be good library and community support and it so happens to be the case with Scala and JRuby. There is more library support and frameworks in Ruby compared to Scala. However Scala is a newer language. The following information were extracted from –

- http://awesome-ruby.com/#awesome-ruby-web-frameworks
- https://www.scala-js.org/libraries/libs.html

Below is a few examples of the library support for each language from the above websites -

	Scala	JRuby
IDE	Yes	Yes
Build Tools	Gradle (Scala Plugin), SBuild, SBT	JBuilder, Grape
Quality Assurance	ScalaMock, Specs2, Specs	RSpec, minitest
Documentation	Colladoc, Scala-collection	Apipie, Doctor
Logging	Logula, zero-log	Cabin, HttpLog, Log4R
Command Line Parsers	Argot, OptParse, AScalaParseClass	GLI, Main, Rake
Software Design	Hammurabi (Rule Engine and DSL)	Grape, Crepe, Her
Serialization	JSON, XML, XMI	JSON, XML, XMI
Data Storage and ORM	NoSQL, SQL, ORM	NoSQL, SQL, ORM
	Memcached, Cassandra, CouchDB	Cassandra, Mongodb
Web		Camping, Ruby on Rails
Frameworks	RESTFUL API, MVC	Ruby on Rails, MVC
Language and Text Processing	Yes	Yes

JRuby uses RubyGems as a package manager that contains a vast amount of libraries to support Ruby development including –

- Ruby on Rails – a web framework
- Rake - a build language, very similar to Gradle, Make or Ant
- Bundler – application management and dependency management.

Scala has various libraries which can be found but there isn't a vast support of libraries compared to JRuby. JRuby is a popular and easy to learn language suited for most software developers today.

Integration with web Technologies

Web technologies are starting to become the defacto for modern software development and these languages have kept up with the times by providing multiple web frameworks such as Model-View-Controller frameworks, web sockets, HTML Templating Language and web DSL Support such as ScalaTags.

Summary

JRuby would be ideal for projects that are small to medium sized internal DSLs that require a quick return of investment from the project. The dynamically type language is simple and less noisy compared to other statically typed languages. They are also ideal to encapsulate fairly simple domain knowledge into a DSL and Projects that require lots of library support and community support. On the other hand Scala would be ideal for Large sized internal DSLs. These do take a longer time to implement and are more complex but they are able to encapsulate complex domain knowledge such as financial models and scientific calculations. They also Work well with the functional facilities provided by Scala. Scala is also ideal when creating external DSLs and projects that will work alongside other DSLs made with Scala.