Fowler's State Machine DSL

H. Conrad Cunningham

16 April 2022

Contents

Professor of Computer and Information Science
University of Mississippi
214 Weir Hall
P.O. Box 1848
University, MS 38677
(662) 915-7396 (dept. office)

Browser Advisory: The HTML version of this textbook requires a browser that supports the display of MathML. A good choice as of April 2022 is a recent version of Firefox from Mozilla.

Fowler's Reader DSLs

Background on DSLs

The State Machine (Secret Panel) DSLs case study is based on the running example in Martin Fowler's book *Domain Specific Languages* [1]. The example is introduced in Chapter 1 and then used in other chapters throughout the book.

- See the article Domain-Specific Languages: An Introductory Example.

Fowler's book defines a pattern language for developing domain-specific languages. His DSL Patterns Catalog is online at http://martinfowler.com/dslCatalog/.

Source Code for DSLs

I originally developed these DSLs in Spring 2009 by referring to an in-work version of Fowler's book on his website, so my examples may not precisely follow the book as published in late 2010.

Fowler's code is written in various languages, including Java, Ruby, and C#. I used Scala 2 [2,4]. I updated the Scala programs in 2018 to make sure they would compile and execute under the then-current Scala release.

2022 Note: I have not updated the code yet for Scala 3 [3,4].

TODO: Update the Scala code to be compatible with Scala 3 without deprecation warnings.

Scala State Machine Semantic Model

This DSL is based, in part, on Fowler's Chapters 1, 3, and 11:

- StateMachine.scala
- StateMachineTest.scala
- CommandChannel.scala (mock)
- StateMachineDirect.sh test script

Scala XML-based External DSL

This DSL is based, in part, on Fowler's Chapters 1, 3, and 5:

- StateMachineXMLTest .scala
- IncrementalStateMachineBuilder.scala
- input file SecretPanel.xml
- StateMachineXML.sh test script

Scala Custom External DSL with Delimiter-Directed Parser

This DSL is based, in part, on Fowler's Chapters 1, 3, 5, and 17:

- DelimiterDSLTest.scala
- IncrementalStateMachineBuilder.scala
- input file CustomExternalStateMachineDSL.dsl
- DelimiterDSL.sh test script

Scala Custom External DSL with Hand-Coded Ad Hoc Recursive-Descent Parser

This DSL is based, in part, on Fowler's Chapters 1, 3, 5, and 21:

- RecursiveDescentTest.scala
- IncrementalStateMachineBuilder.scala
- input file CustomExternalStateMachineDSL2.dsl
- RecursiveDescent.sh test script

Scala Custom External DSL (using Scala Parser Combinator Library) with Embedded State Machine Builder

This DSL is based, in part, on Fowler's Chapters 1, 3, 5, and 22:

- CombinatorParserBuilderTest.scala
- IncrementalStateMachineBuilder.scala
- input file CustomExternalStateMachineDSL2.dsl
- CombinatorParserBuilder.sh test script

Scala Custom External DSL (using Scala Parser Combinator Library) with Full AST Construction

This DSL is based, in part, on Fowler's Chapters 1, 3, 5, and 22:

- CombinatorParserASTTest.scala
- input file CustomExternalStateMachineDSL2.dsl
- CombinatorParserAST.sh test script

Scala Static C Code Generator with Model-Aware Target Platform Library

This DSL is based, in part, on Fowler's Chapters Fowler Ch. 1, 3, 5, 8, 52, and 55:

- StaticC_GeneratorTest.scala
- StaticC_Generator.sh test script
- generated C output file output.c

Note: The needed framework code needed to run the generated C program is not yet available in this form. It needs to be reconstructed from Fowler's book.

Scala Graphviz Dot Language Code Generator

This DSL is based, in part, on Fowler's Chapters Fowler Ch. 1, 3, 5, 8, and 52:

- GraphVizCodeGen.scala
- GraphVizCodeGenTest.scala
- GraphVizCodeGen.sh test script
- generated Graphviz Dot output file graph.gv — See test script.
- Scalable Vector Graphics output from dot graph.svg — See test script.
- PDF output from dot graph.pdf — See test script.

Note: I developed this DSL in Spring 2018. It requires installation of the GraphViz graphics DSL processor [6] (i.e., DOT [5,7] to generate the graphs.

Acknowldgements

I developed the State Machine (Secret Panel) DSLs case study for my prototype offering of the graduate Software Language Engineering class (eventually CSci 658) in Spring 2009. I based it on the running example in Martin Fowler's book *Domain Specific Languages* [1]. The example is introduced in Chapter 1 (Domain-Specific Languages: An Introductory Example) and then used in other chapters throughout the book.

I originally developed these DSLs in Spring 2009 by referring to an in-work version of Fowler's book on his website, so my examples may not precisely follow the book as published in late 2010.

Fowler's code is written in various languages, including Java, Ruby, and C#. I used Scala 2 [2,4].

In 2018, I updated the DSL implementations slightly to make sure all were compatible with the then-current release of Scala 2. I also added "code generator" for GraphViz.

I retired from the full-time faculty in May 2019. As one of my post-retirement projects, I am continuing work on possible textbooks based on the course materials I had developed during my three decades as a faculty member. In January 2022, I began refining the existing content, integrating separately developed materials together, reformatting the documents, constructing a unified bibliography (e.g., using citeproc), and improving my build workflow and use of Pandoc. I adapted this index page from a portion of my Spring 2018 course notes.

The Scala code for these DSLs have not yet been updated for Scala 3 [3,4].

I maintain this chapter as text in Pandoc's dialect of Markdown using embedded LaTeX markup for the mathematical formulas and then translate the document to HTML, PDF, and other forms as needed.

References

[1] Martin Fowler and Rebecca Parsons. 2010. *Domain specific languages.* Addison-Wesley, Boston, Massachusetts, USA.

[2] Martin Odersky, Lex Spoon, and Bill Venners. 2008. *Programming in Scala* (First ed.). Artima, Inc., Walnut Creek, California, USA.

[3] Martin Odersky, Lex Spoon, and Bill Venners. 2021. *Programming in Scala* (Fifth ed.). Artima, Inc., Walnut Creek, California, USA.

[4] Scala Language Organization. 2022. The Scala programming language. Retrieved from https://www.scala-lang.org/

[5] The Graphviz Authors. 2022. DOT Language. Retrieved from https://graphviz.org/doc/info/lang.html

[6] The Graphviz Authors. 2022. Graphviz. Retrieved from https://learnxinyminutes.com/docs/haskell/

[7] Wikpedia: The Free Encyclopedia. 2022. DOT (graph description language). Retrieved from https://en.wikipedia.org/wiki/DOT_(graph_description_language)