

RDF Coder

Modeling Java with RDF

Purpose

RDFCoder is a flexible tool meant to convert Java code, both Source or Bytecode, to an RDF model representation.

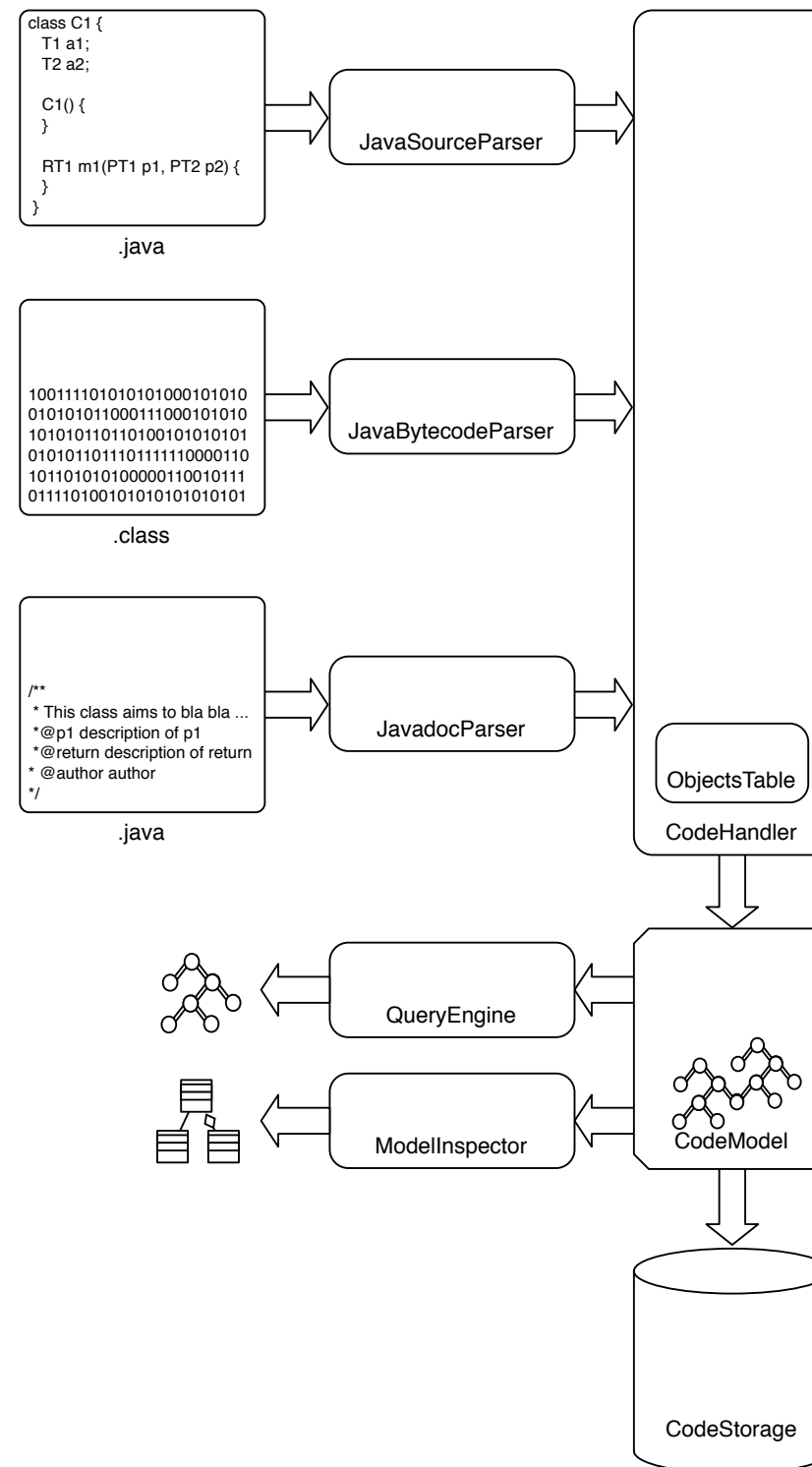
How RDF Coder can be useful for?

Code Analysis - Analyze package structures, class fields, methods, signatures, library dependencies.

Code Refactoring - Find library problems, find separation points among libraries.

Custom Documentation - Generate your custom documentation starting from the model.

Overall Architecture



Components (I)

- **Java Source Parser** - a set of classes able to parse .java files and convert its contents in CodeHandler events.
- **Java Bytecode Parser** - a set of classes able to parse .class files and convert its contents in CodeHandler events.
- **Javadoc Parser** - a set of classes able to parse .java files Javadoc documentation and convert it in CodeHandler events.
- **Code Handler** - the Code Handler is a class able to convert class events in RDF triples. The way the Code handler translates received events into triples is described by the Kabbala Model.
- **Objects Table** - the Code Handler needs to fully qualify (i.e.: add full package qualification) all the objects it finds during the compilation process. The fully qualification is done by populating and querying the Objects Table. Every time that an Object needs to be fully qualified, the Objects Table is inquired by providing the object name and the Imports Context that is the list of the imports provided when used the object itself.

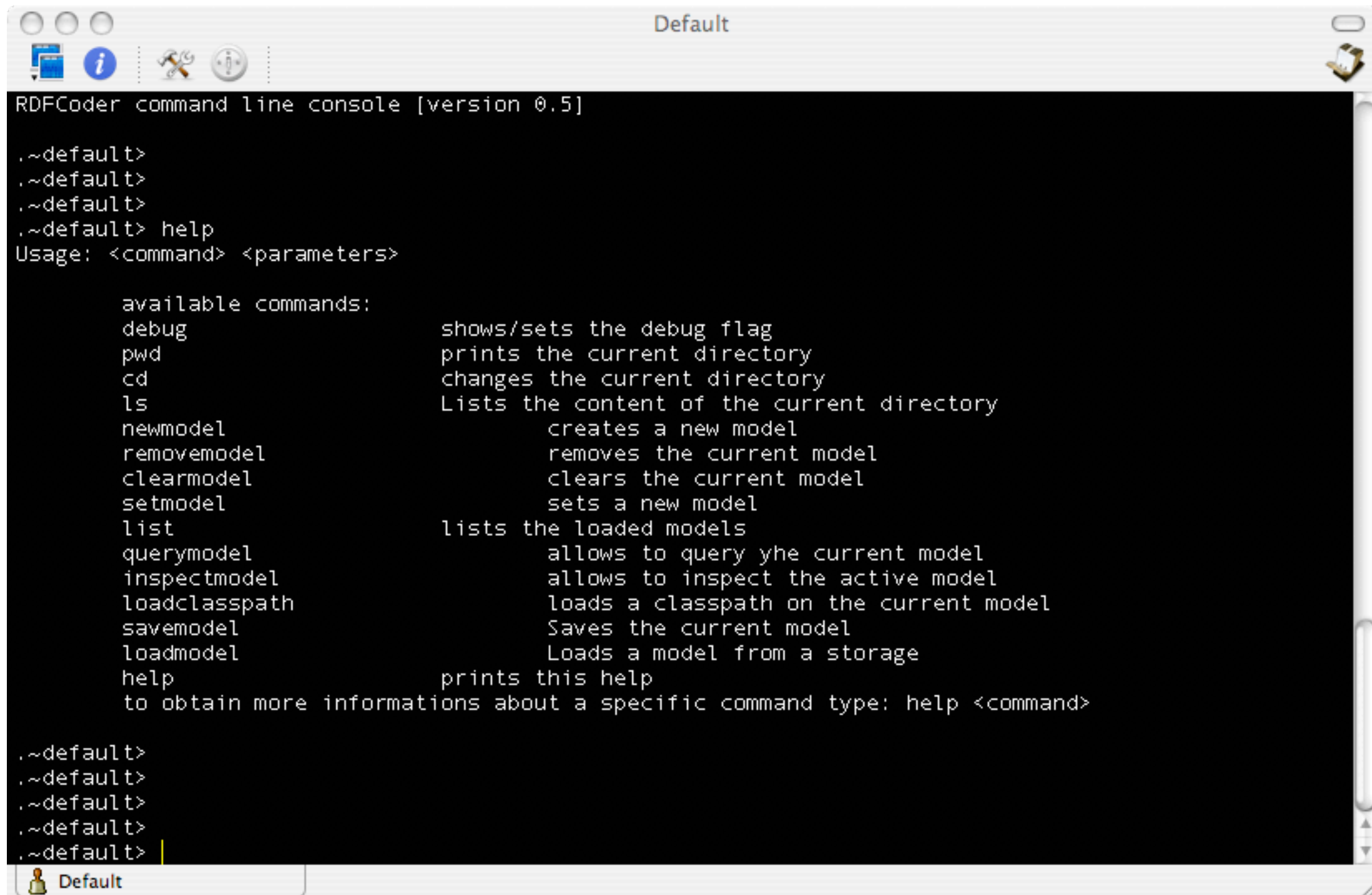
Components (2)

- **Code Model** - the Code Model is a container of the triples representing a bunch of related libraries. A code model provides operations to add triples, remove triples, perform basic search on triples, perform complex queries.
- **Code Storage** - the Code Storage is a set of classes meant to make persistent a code model. There may be several persistent storages like Filesystem or a RDMS.
- **Query Engine** - the Query Engine is a set of classes meant to perform queries on a storage. At the moment the only supported query language is SPARQL.
- **Model Inspector** - the Model Inspector is a high level representation of the CodeModel. By using the Model Inspector it is possible to navigate Java objects representing the entities stored into the model.

Command line (I)

- **debug** Enables / disables the debug mode.
- **pwd** Prints the current directory.
- **cd** Changes the current directory.
- **ls** Lists the content of the current directory.
- **list** Lists all the currently loaded models.
- **setmodel** Allows to set the selected model.
- **newmodel** Creates a new Code Model to store library informations.
- **clearmodel** Cleans the content of a Code Model.
- **removemodel** Removes a previously created Code Model.
- **querymodel** Allows to perform a query (SPARQL) on the current model.
- **inspectmodel** Allows to inspect the current model.
- **loadmodel** Loads a library model.
- **savemodel** Saves a model on a storage.
- **loadclasspath** Loads source classpath extracting the RDF representation.

Command line (2)



The screenshot shows a macOS-style window titled "Default" with a toolbar containing icons for file operations, information, and settings. The main area is a black terminal window titled "RDFCoder command line console [version 0.5]". It displays the following text:

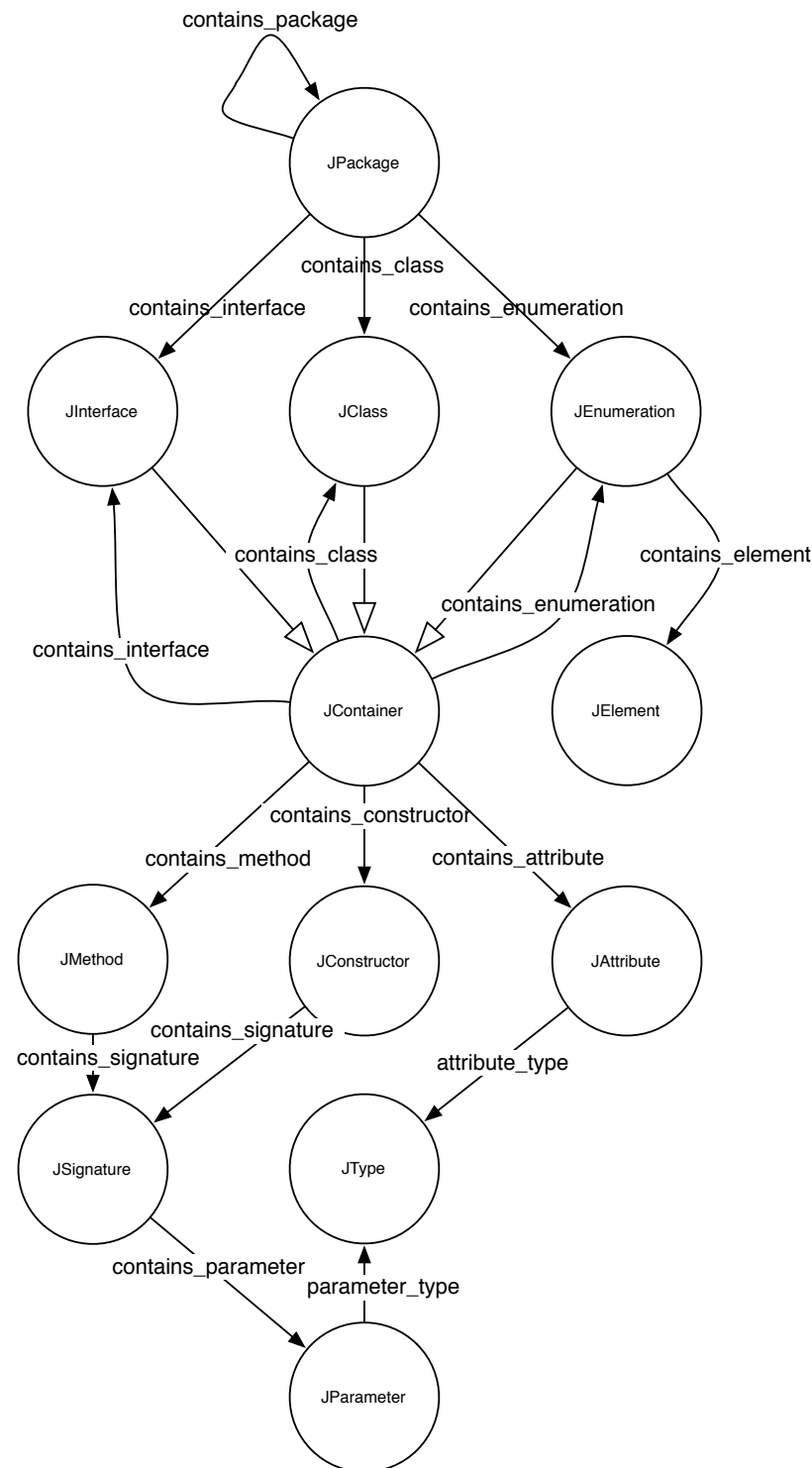
```
~default>
~default>
~default>
~default> help
Usage: <command> <parameters>

    available commands:
    debug                shows/sets the debug flag
    pwd                  prints the current directory
    cd                   changes the current directory
    ls                   Lists the content of the current directory
    newmodel              creates a new model
    removemodel           removes the current model
    clearmodel            clears the current model
    setmodel              sets a new model
    list                  lists the loaded models
    querymodel            allows to query yhe current model
    inspectmodel          allows to inspect the active model
    loadclasspath         loads a classpath on the current model
    savemodel             Saves the current model
    loadmodel             Loads a model from a storage
    help                  prints this help
    to obtain more informations about a specific command type: help <command>

~default>
~default>
~default>
~default>
~default> |
```

At the bottom of the window, there is a status bar with a person icon and the text "Default".

The Kabbalah Model



Status

- Version 0.5
- Still under debugging
- Stable version 0.6

Next Features

- Extend it to cross language support.

References

<http://rdfcoder.googlecode.com/svn>