jInfer Base Module Description

Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, Matej Vitásek
Advisors: RNDr. Irena Mlýnková, Ph.D., Martin Nečaský, Ph.D.

Praha, 2011

Target audience: developers willing to extend jInfer.

Responsible developer	Matej Vitásek
Required tokens	none
Provided tokens	none
Module dependencies	none

Public packages	`cz.cuni.mff.ksi.jinfer.base.automaton`
	`cz.cuni.mff.ksi.jinfer.base.interfaces`
	`cz.cuni.mff.ksi.jinfer.base.interfaces.inference`
	`cz.cuni.mff.ksi.jinfer.base.interfaces.nodes`
	`cz.cuni.mff.ksi.jinfer.base.objects`
	`cz.cuni.mff.ksi.jinfer.base.objects.nodes`
	`cz.cuni.mff.ksi.jinfer.base.regexp`
	`cz.cuni.mff.ksi.jinfer.base.utils`
	`org.apache.log4j.*`

1 Introduction

This is the module containing data structures, interfaces and logic shared across the whole jInfer framework. Virtually every other module containing logic should in theory depend on *Base*.

2 Structure

Description of *Base* structure will partially mirror its JavaDoc documentation ([KMS+g]). For more detailed information, refer to it directly.

Base contains logic for Log4j ([log]) initialization in `Installer` class. Configuration of the overall logging granularity level (NetBeans options integration) is contained in `cz.cuni.mff.ksi.jinfer.base.options` package. Shared jInfer graphics is contained in the `cz.cuni.mff.ksi.jinfer.base.graphics` and `cz.cuni.mff.ksi.jinfer.base.graphics.icons` packages.

2.1 Data structures

Regular expression representation is contained in `cz.cuni.mff.ksi.jinfer.base.regexp` package. Most important class is of course `Regexp`, assisted by an enum of its type `RegexpType` and representation of its interval `RegexpInterval`.
XML node representation is described by interfaces in `cz.cuni.mff.ksi.jinfer.base.interfaces.nodes` package and more-less concrete implementations in `cz.cuni.mff.ksi.jinfer.base.objects.nodes` package.
Finite state automata representation is contained in the `cz.cuni.mff.ksi.jinfer.base.automaton` package.
Refer to [KMS+a] to get an accurate description of all these representations.

Miscellaneous shared classes are contained in `cz.cuni.mff.ksi.jinfer.base.objects`. For example, `Input` is used to provide the *Initial Grammar Generator* with input data. `Pair` is a generic class binding two object together in a *pair*.

2.2 Interfaces

Apart from interfaces contained in already discussed `cz.cuni.mff.ksi.jinfer.base.interfaces.nodes` package, there is an important group of interfaces contained in `cz.cuni.mff.ksi.jinfer.base.interfaces.inference` package: the *inference* interfaces. They come in two flavours: the actual inference interface, and its callback. For a comprehensive description of them and their interaction refer again to [KMS$^+$a].

There is one more package containing interfaces: `cz.cuni.mff.ksi.jinfer.base.interfaces`. There are a few groups of them.

- Module lookup support: `NamedModule` and `UserModuleDescription`.
- Inference support: `Capabilities`.
- Service provider definitions: `Expander`, `Processor` and `RuleDisplayer`.

2.3 Utility logic

Useful logic for the whole framework is focused in the `cz.cuni.mff.ksi.jinfer.base.utils` package. A few highlights from here follow.

- Testing whether a collection is empty: `BaseUtils.isEmpty()` - handles `null` and zero elements.
- Filtering a list based on a predicate: `BaseUtils.filter()`.
- Cloning a list N times in a row: `baseUtils.cloneList()` - e.g. from *abc, 3 times* creates *abcabcabc*.
- Deep cloning a grammar: `CloneHelper.cloneGrammar()`.
- Writing out a list separated by specified separator: `CollectionToString.colToString()` - accepts the list, operation to perform on each element and separator character. Separator is placed smartly, i.e. only between elements.
- XML element comparison while ignoring specified members: `EqualityUtils`.
- Module lookup/selection: `ModuleSelectionHelper`.
- Singleton class reporting on the currently running inference: `RunningProject`.
- Various utilities for JUnit tests: `TestUtils`.
- Topological sorting of grammar: `TopologicalSort`.

3 Data flow

This is not an inference module and there is no real data flow in it. Refer to [KMS$^+$a] to understand the inference process described by interfaces from `cz.cuni.mff.ksi.jinfer.base.interfaces.inference` package.

4 Extensibility

From one point of view, extensibility of *`Base`* means creating service providers implementing inference or other interfaces. This is described in the documentation for respective modules that use these interfaces.
On the other hand, if there is a need to share data structures, interfaces or logic across multiple developed modules (for example between custom schema generator and simplifier), it is advised to create a module similar to *`Base`* instead of changing it directly.

References

[Aho96] H. Ahonen. *Generating grammars for structured documents using grammatical inference methods*. PhD thesis, Department of Computer Science, University of Helsinki, Series of Publications A, Report A-1996-4, 1996.

[BNST06] Geert Jan Bex, Frank Neven, Thomas Schwentick, and Karl Tuyls. Inference of concise dtds from xml data. In *Proceedings of the 32nd international conference on Very large data bases*, VLDB '06, pages 115–126. VLDB Endowment, 2006.

[BNV07] Geert Jan Bex, Frank Neven, and Stijn Vansummeren. Inferring xml schema definitions from xml data. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 998–1009. VLDB Endowment, 2007.

[Bou] Ronald Bourret. Dtd parser, version 2.0. `http://www.rpbourret.com/dtdparser/index.htm`.

[gra] Graph visualization software. `http://www.graphviz.org/`.

[HMU01] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (2nd Edition)*. Addison-Wesley, 2001.

[HW07] Yo-Sub Han and Derick Wood. Obtaining shorter regular expressions from finite-state automata. *Theor. Comput. Sci.*, 370(1-3):110–120, 2007.

[JAX] Java architecture for xml binding. `http://jaxb.java.net/`.

[jun] Java universal network/graph framework. `http://jung.sourceforge.net/`.

[KMS$^+$a] Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, and Matej Vitásek. *jInfer Architecture*.

[KMS$^+$b] Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, and Matej Vitásek. *jInfer AutoEditor automaton visualization and editor module*.

[KMS$^+$c] Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, and Matej Vitásek. *jInfer Base Module Description*.

[KMS$^+$d] Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, and Matej Vitásek. *jInfer BasicDTDExporter Module Description*.

[KMS$^+$e] Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, and Matej Vitásek. *jInfer BasicIGG Module Description*.

[KMS$^+$f] Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, and Matej Vitásek. *jInfer BasicRuleDisplayer Module Description*.

[KMS$^+$g] Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, and Matej Vitásek. jinfer javadoc. `http://jinfer.sourceforge.net/javadoc`.

[KMS$^+$h] Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, and Matej Vitásek. *jInfer TwoStep simplifier design and implementation*.

[log] Apache log4j^{TM}. `http://logging.apache.org/log4j/`.

[loo] org.openide.util.class lookup. `http://bits.netbeans.org/dev/javadoc/org-openide-modules/org/openide/modules/doc-files/api.html`.

[mod] Module system api. `http://bits.netbeans.org/dev/javadoc/org-openide-modules/org/openide/modules/doc-files/api.html`.

[net] Todo. `http://netbeans.org`.

[Nor] Theodore Norvell. A short introduction to regular expressions and context free grammars. `http://www.engr.mun.ca/~theo/Courses/fm/pub/context-free.pdf`.

[pro] Project sample tutorial. `http://platform.netbeans.org/tutorials/nbm-projectsamples.html`.

[VMP08] Ondřej Vošta, Irena Mlýnková, and Jaroslav Pokorný. Even an ant can create an xsd. In *DASFAA'08: Proceedings of the 13th international conference on Database systems for advanced applications*, pages 35–50, Berlin, Heidelberg, 2008. Springer-Verlag.

[Vyh] Julie Vyhnanovská. Automatic construction of an xml schema for a given set of xml documents.

[wik] Regular expression. `http://en.wikipedia.org/wiki/Regular_expression`.

[xml] Xml validation api. `http://download.oracle.com/docs/cd/E17802_01/webservices/webservices/docs/1.6/api/javax/xml/validation/package-summary.html`.