jInfer ProjectType 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, specifically extend jInfer project structure.

Responsible developer:	Michal Švirec
Required tokens:	cz.cuni.mff.ksi.jinfer.base.interfaces.inference.IGGenerator cz.cuni.mff.ksi.jinfer.base.interfaces.inference.SchemaGenerator cz.cuni.mff.ksi.jinfer.base.interfaces.inference.Simplifier org.openide.windows.IOProvider
Provided tokens:	none
Module dependencies:	Base Runner
Public packages:	cz.cuni.mff.ksi.jinfer.projecttype.actions

1 Introduction

ProjectType is the module responsible for creation of NBP project type which groups input/output files that belongs to one specific inference. Each jInfer project also allows user to set properties specific for inference.

With each project created in jInfer is also created specific directory structure on filesystem that describes jInfer project. This structure is described in figure 1. Folder in filesystem is considered as jInfer project folder, if contains *jinferproject* subfolder. This folder contains two files, first is *project.properties*, where all properties set for particular project are saved. Second file is *input.files* which is XML file filled with paths to files inserted as input into particular project (structure of this file is described in section 2.1.3). *Output* subfolder contains schema files which were generated by jInfer inference.

2 Structure

Structure of *ProjectType* can be divided into following five main parts.

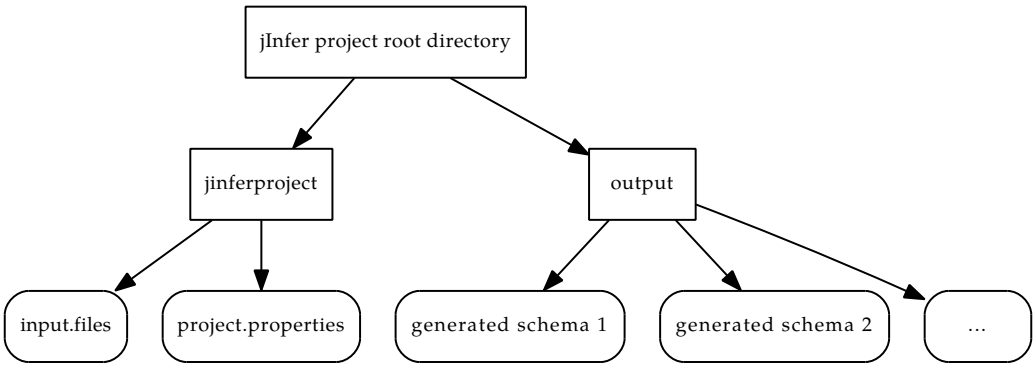

Figure 1: Project directory structure

- Base classes - Classes providing main functionality like creation of project, defining operations like move, delete, copy etc. All the base classes are contained in the `cz.cuni.mff.ksi.jinfer.projecttype` package.
- Visualization classes - These classes create tree structure of project in NBP Projects view and are contained in the `cz.cuni.mff.ksi.jinfer.projecttype.nodes`
- Actions - Classes from `cz.cuni.mff.ksi.jinfer.projecttype.actions` package that provides actions allowing adding, removing input files into project, or running the project.
- Properties - Classes responsible for creation of project properties window with properties category tree. These are situated in `cz.cuni.mff.ksi.jinfer.projecttype.properties` package.
- Project wizard - Classes representing creation of project through new project wizard from `cz.cuni.mff.ksi.jinfer.projecttype.sample` package.

2.1 Base classes

This section describes classes needed for creation of the jInfer project and correct integration into NBP. Main two classes representing jInfer project type are `JInferProjectFactory` and `JInferProject`.

2.1.1 JInferProjectFactory

`JInferProjectFactory` is a factory class implementing `ProjectFactory` interface provided by NBP. This class is responsible for loading/saving of jInfer project from/to filesystem and also determining if folder in filesystem is type of jInfer project (contains *jinferproject* subfolder). For this purpose, class has three methods: `isProject()`, `save()` and `load()`. When jInfer project is loaded from filesystem, `JInferProjectFactory` creates new instance of `JInferProject` passing path to project folder as a constructor parameter. When `save()` method is invoced, all the project properties are saved in *projectproperties* file and paths to input files are saved in *input.files*.

2.1.2 JInferProject

`JinferProject` class implements `Project` interface from NBP and represents in-memory representation of jInfer project. Inner structure of this class is very simple, interface provides only two methods: `getProjectDirectory()` and `getLookup()`. All the extensibility of project type is done by *Lookup* functionality of NBP. Project lookup contains besides properties and `Input` class, which encapsulates input files, also class that creates output files, class that builds project tree in project window, etc.

2.1.3 InputFiles class

`InputFiles` is utility class that stores/loads input file paths into/from input.files XML file. For this purpose, class uses *JAXB* [JAX] architecture to unmarshall/marshall XML file into/from java content objects. XML structure of the input.files is very simple, *jinferinput* is root element under which are *documents*, *schemas* and *queries* elements. Each of this elements could have none or more *file* elements with string attribute *loc* that contains absolute path to input file.

Example of the structure follows.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<jinferinput>
    <xml>
        <file loc="/home/user/example.xml"/>
    </xml>
    <schemas>
        <file loc="/home/user/example.dtd"/>
    </schemas>
    </queries/>
</jinferinput>
```

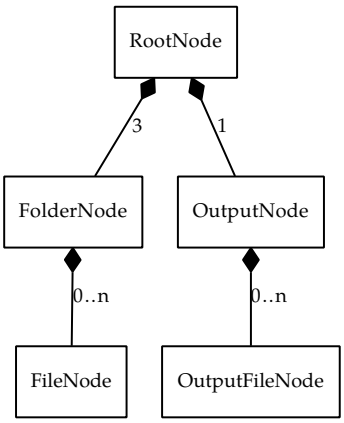

Figure 2: Project directory structure

2.2 Visualization classes

Each project in NBP is after creation opened in project window and has some visual tree structure where each node in this tree is represented by NBP `Node` class. This structure is in jInfer project type provided by `JInferLogicalView` class that implements `LogicalViewProvider` interface from NBP and is available through `JInferProject` lookup. This class has one important method, `createLogicalView()` that returns instance of `RootNode` class of root node. Simple class structure is described in figure 2.

`RootNode` extends NBP `AbstractNode` class which is basic implementation of NBP `Node`. This node contains four children nodes: three for input files (Document, Schema and Query) and one for output files generated by inference (Output). Root node also offers actions for adding input files, run inference and standard project actions like delete or move/rename project.

Each of the *input* nodes is represented by `FolderNode` class and contains *input file* nodes (represented by `FileNode` class) representing files added to the jInfer project. Notice that these *input file* nodes represents files from filesystem and are not copied to project folder structure. Because of this, if you delete this nodes from project tree, files from filesystem are not deleted.

`OutputNode` class represents *Output* node in jInfer project type tree and is slightly different from *input* nodes, because it refers to *output* subfolder in project filesystem hierarchy. In other words, files contained in this folder are shown as subnodes in *Output* node in project tree. This subnodes are represented by `OutputFileNode` class. Unlike *input file* nodes, if you delete *output file* node under the *Output* node, file represented by this node is also deleted from filesystem.

2.3 Actions

Each node in project tree hierarchy contains set of actions available from its context menu. Besides standard actions for root node or file node, jInfer project type introduce its own actions. These actions could be separated into three groups: actions for *root* node, actions for *input folder* nodes and one action for *input/output file* nodes implemented by `ValidateAction` class. All classes implementing some jInfer project action extend standard java `javax.swing.AbstractAction` class besides `ValidateAction` class which is described in 2.3.3.

2.3.1 Root node actions

This group contains two actions, first is action that allows adding input files into *input folder* nodes (implemented by `FilesAddAction` class). Second one is slightly different, because unlike from the previous action, this one doesn't create new item in *root* node context menu but implements functionality for item that already exists in context menu - Run action.

First of the two actions, after it was invoked, open file choose dialog which has set file filter according to available classes that implement `Processor` interface registered in jInfer with `ServiceProvider`. After selecting files in dialog, these files are inserted into particular *input file* node according to mapping of file extension to folder type gained from `Processor` classes.

Run action, which is available in root node context menu, is also present in NBP toolbar as a *Run* button and in jInfer Welcome window. This action is implemented in `RunAction` class and simply invokes *`Runner`*`s run()` method, if no other inference is already running, otherwise information dialog pops up informing about already running inference. This check is done with `setActiveProject()` from `RunningProject` class.

2.3.2 Input folder node actions

To this group of actions belong two actions: action for adding input files into *input folder* node (`FileAddAction` class) and action for deleting all input files in *input folder* node (`DeleteAllAction` class). `FileAddAction` class implements action which opens file choose dialog and selected files are added into particular *file input*node. `DeleteAllAction` class simply removes all input files previuosly added into particular *file input* node.

2.3.3 ValidateAction class

`ValidateAction` class implements action, which allows documents and schema files to be validated against each other. Because of this, action is presented in context menu of document and schema files (in actual implementation is available for XML, dtd and XML schema files). Unlike the other action classes, this extends NBP `NodeAction` class, which allows to enable/disable action in context menu according to node selection. Action is enabled only if exactly one schema file is selected and one or more XML files. If the selected XML files are valid against selected schema, information dialog is popped up, otherwise error dialog is popped up and in *validate* output window is printed error message describing cause of non-validity XML files against schema.

2.4 Properties

Each jInfer project has associated properties window through which can user change project-wide properties. This window is accesible through *properties* action in project *root* node context menu. Properties window consists of category tree on a left side and properties pane on a right side. For each category in tree is associated one properties pane. Classes responsible for creation of this properties window are `JInferCustomizerProvider` and `JInferComponentProvider`.

2.5 Project wizard

2.6 Preferences

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.

[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™. `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`.

[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`.

[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 Vyhňanovská. Automatic construction of an xml schema for a given set of xml documents.

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