http://kieker.sourceforge.net

Kieker Developer Guide

Kieker Project

March 18, 2011

Software Engineering Group
University of Kiel, Germany

Contents

1 Introduction

2 Project Management

2.1 Subversion Repository

We provide anonymous (read-only) access to the Kieker subversion repository:

- **URL:**
  `https://samoa.informatik.uni-kiel.de/svn/projects/kieker/software/kieker/`
- **Username:** anonymous
- **Password:** anonymous

In order to request write access, please send an e-mail to `avh@informatik.uni-kiel.de`.

2.2 Mailing Lists

You should register to the following to mailing lists:

- **kieker-users@sourceforge.net:**
  `https://lists.sourceforge.net/lists/listinfo/kieker-users`
- **kieker-developers@sourceforge.net:**
  `https://lists.sourceforge.net/lists/listinfo/kieker-developers`

2.3 Bug Tracker

The bug tracker can be accessed via:

`http://sourceforge.net/tracker/?group_id=212691&atid=1022738`

3 Build and Development Environment

3.1 Ant Build

Ant is used as the build tool for Kieker. The important ant targets included in the `build.xml` file are:

- **build-all:** This is the default target that compiles the Kieker artifacts and creates the following jar/war files in the `dist/` directory:
  - `kieker-analysis-1.5-trunk.jar`
  - `kieker-common-1.5-trunk.jar`
  - `kieker-monitoring-1.5-trunk.jar`
  - `kieker-monitoring-servlet-1.5-trunk.war`
  - `kieker-tools-1.5-trunk.jar`
- **javadoc:** Executes Javadoc to create the API documentation in HTML. After a successful execution, open `build/javadoc/index.html` in your Web browser.
- **run-tests-junit:** Runs the JUnit tests and creates a HTML report of the results.
- **run-test**[s]**-*:** Additional regression tests mostly employing AspectJ for instrumentation.
- **release:** Creates the following Kieker release files in the `dist/release/` directory:
  - `kieker-<version>_javadoc.(zip|tar.gz)`
  - `kieker-<version>_binaries.(zip|tar.gz)`
  - `kieker-<version>_sources.(zip|tar.gz)`
  - `kieker-<version>_examples-JPetStoreExample.(zip|tar.gz)`
  - `kieker-<version>_examples-MySimpleKiekerAspectJExample.(zip|tar.gz)`
  - `kieker-<version>_examples-MySimpleKiekerJMSExample.(zip|tar.gz)`
  - `kieker-<version>_examples-OverheadEvaluationMicrobenchmark.(zip|tar.gz)`

3.2 IDE-Specific Project Configuration

This section describes how to create a project for developing with the Kieker sources in different IDEs.

3.2.1 Netbeans

You create a Netbeans project for the Kieker trunk as follows:

1. File->New Project... and select *Java Free-Form Project* and follow the wizard and answer the dialogs as follows:
   a) **Name and Location (Fig. 3.1):**
   Select Kieker's `trunk/` directory as the project location and th wizard will complete the missing information based on the existing `build.xml` file.

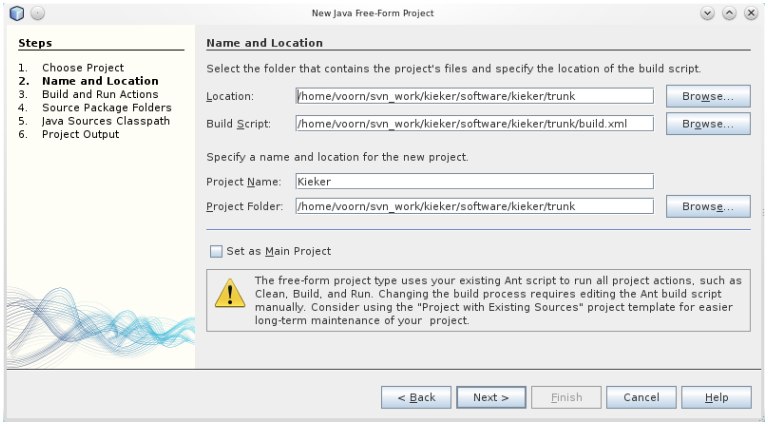

Figure 3.1

   b) **Build and Run Actions (Fig. 3.2):**
   Complete the information as shown in Fig. 3.2.

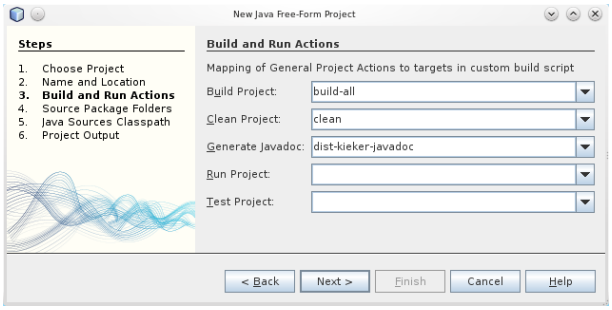

Figure 3.2

   c) **Source Package Folders (Fig. 3.3):**
   Add alls subdirectories of `src/` and all subdirectories of `test/` (except for `test/META-INF/`) to the list of *Source Package Folders*. **Remove all *Test Package Folders* entries.**

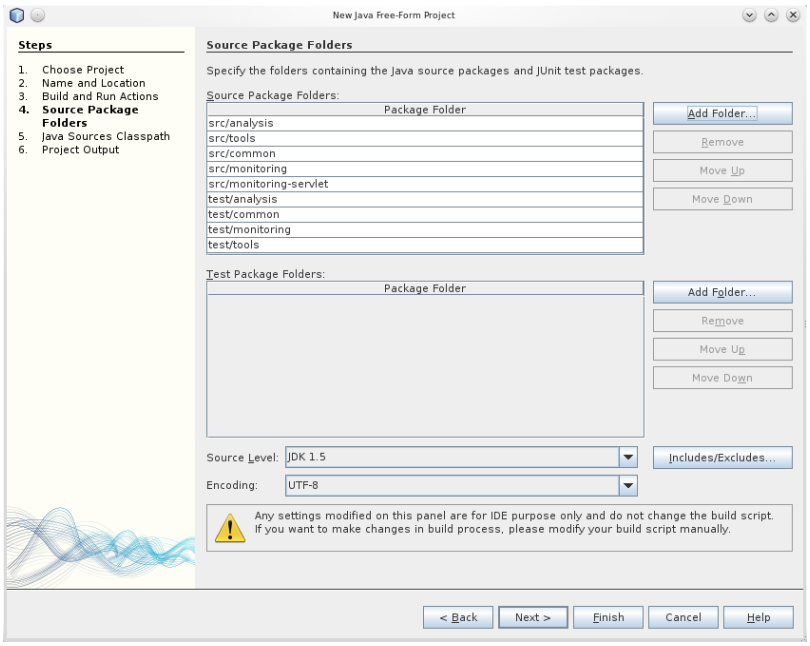

Figure 3.3

d) **Java Sources Classpath (Fig. 3.4):**
Uncheck the option *Separate Classpath for Each Source Package Folder* and **add** all .jar files included in the `lib/` directory.

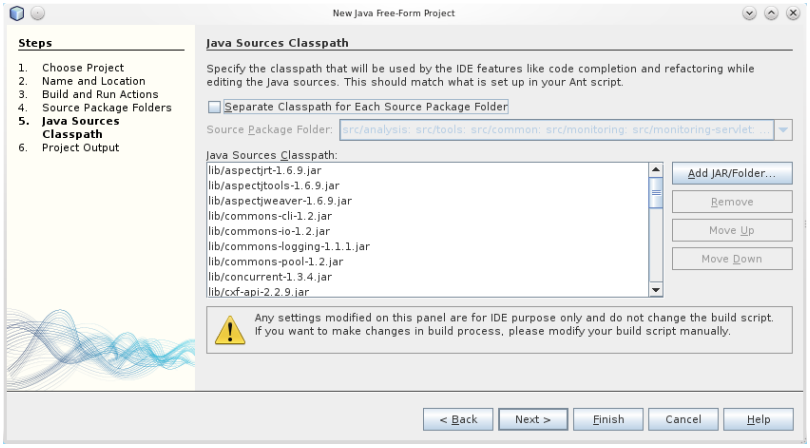

Figure 3.4

e) **Project Output (Fig. 3.5)**
Simply skip this dialog without any changes and finish the wizard by clicking

on the **Finish** button.

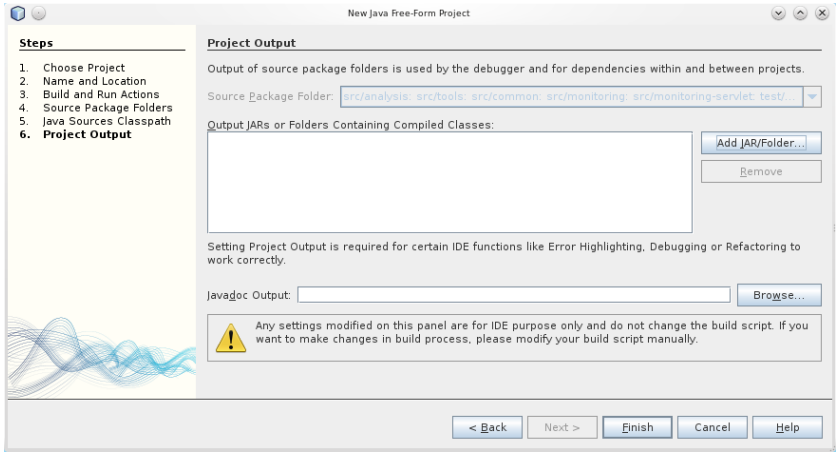

Figure 3.5

f) Kieker should now appear in the list of projects, as shown in Fig. 3.6.

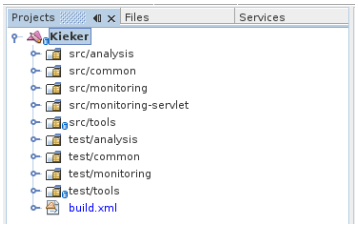

Figure 3.6

3.2.2 Eclipse

You create an Eclipse project for the Kieker trunk as follows:

1. Kieker includes two sample files to create an Kieker Eclipse project:
   - `eclipse-classpath.sample`
   - `eclipse-project.sample`

   Copy these files to `.classpath` and `.project` respectively.
2. Open the Java project creation wizard by selecting File->New->Java Project.
   a) **Create a Java Project (Fig. 3.7):**
   Uncheck *Use default location* and select Kieker's `trunk/` directory from your filesystem. Rename the project to "Kieker".

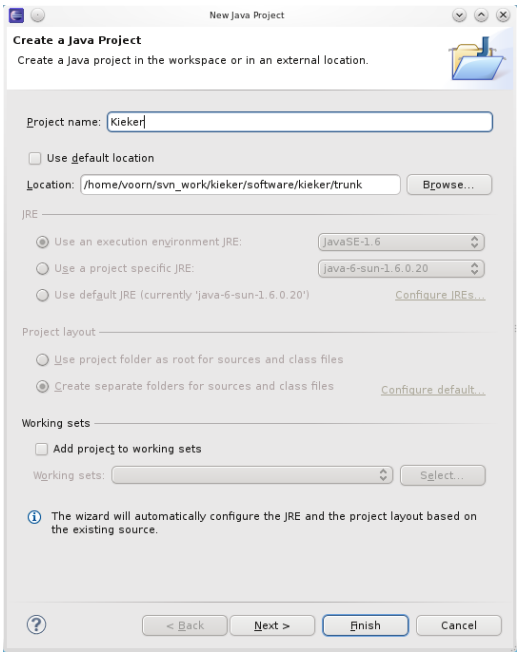

Figure 3.7

b) **Java Settings (Fig. 3.8):**
All settings in this dialog should be imported correctly from the `.project` and `.classpath` files. Simply skip this dialog without any changes and finish the wizard by clicking on the **Finish** button.

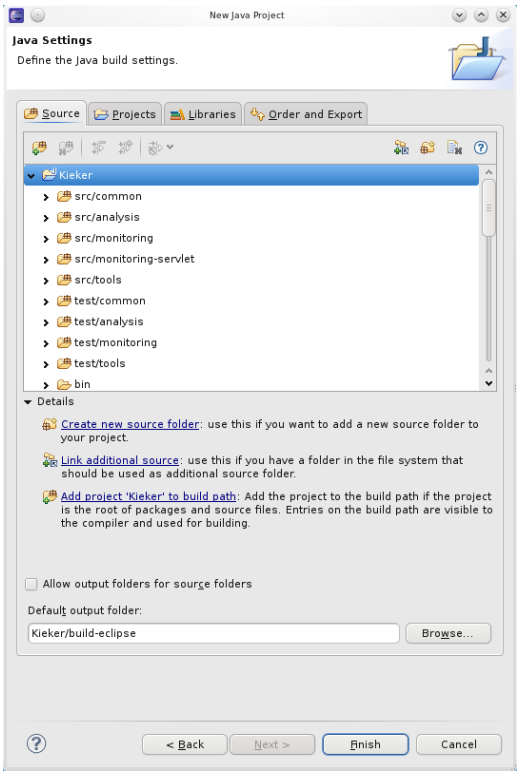

Figure 3.8

c) Kieker should now appear in the list of projects, as shown in Fig. 3.9.

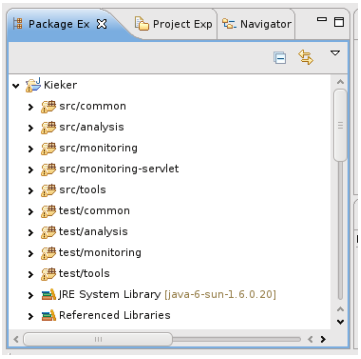

Figure 3.9

3.3 Continuous Integration

Not yet performned, but we might evaluate the following tools

- CruiseControl: `http://cruisecontrol.sourceforge.net/`
- Hudson: `http://hudson-ci.org/`

4 Creating a Release

Checklist for creating a release version

1. Tag the `trunk/` to work on a frozen copy:

```
svn cp trunk/  tags/<version>/
```

2. Set the final release version number in the `build.xml`
3. Modify the `HISTORY` file
4. Commit the tag
5. Export a clean version of the tag from the svn:

```
svn export \
 http:// samoa.informatik.uni-kiel.de /[ ... ]/tags/1.2 kieker-1.2-export \
 --username=USER --password=PASS
```

6. **Build the release files**

```
ant release
```

7. **Test the source release:** — *Extract source release archive to a tmp directory*
   a) Manually inspect contents
   b) Test build:

```
ant
```

   c) Execute JUnit tests

```
ant run-tests-junit
```

   d) Execute integration tests (each test eventually terminates; some take quite a while)

```
ant run-tests-compileTimeWeaving-bookstore
ant run-tests-compileTimeWeaving-bookstore-synchronized
ant run-tests-compileTimeWeaving-bookstoreDB
ant run-tests-compileTimeWeaving-twoConcurrentMethodsExample
```

```
ant run-tests-loadTimeWeaving-bookstoreDifferentRecordTypes
ant run-tests-loadTimeWeaving-bookstoreFullInstrumentation
ant run-tests-loadTimeWeaving-executionOrderTest
ant run-test-storage
```

8. **Test the binary release:** — *Extract binary release archive to a tmp directory*
   a) Test wrapper scripts: — ***Linux & Windows***
   - `bin/convertLoggingTimestamp.sh` — *inspect generated `.html/.txt` files*

```
bin/convertLoggingTimestamp.sh \
 --timestamps 1283156545581511026 1283156546127117246
```

   - `bin/trace-analysis.sh`

```
bin/trace-analysis.sh --inputdirs examples/userguide/ch5--trace-monitoring-aspectj/
    testdata/tpmon-20100830-082225522-UTC/ --outputdir .
    --plot-Deployment-Component-Dependency-Graph
    --plot-Assembly-Component-Dependency-Graph
    --plot-Container-Dependency-Graph
    --plot-Deployment-Operation-Dependency-Graph
    --plot-Assembly-Operation-Dependency-Graph
    --plot-Aggregated-Deployment-Call-Tree --plot-Aggregated-Assembly-Call-Tree
    --print-Deployment-Equivalence-Classes --print-Assembly-Equivalence-Classes
    --plot-Aggregated-Deployment-Call-Tree --plot-Aggregated-Assembly-Call-Tree
    --short-labels

bin/trace-analysis.sh --inputdirs examples/userguide/ch5--trace-monitoring-aspectj/
    testdata/tpmon-20100830-082225522-UTC/ --outputdir . --select-traces
    6488138950668976141 6488138950668976129 6488138950668976130 6488138950668976131
     --plot-Deployment-Sequence-Diagrams --plot-Assembly-Sequence-Diagrams
    --plot-Call-Trees --print-Message-Traces --print-Execution-Traces
    --short-labels
```

   - `bin/dotPic-fileConverter.sh`

```
bin/dotPic-fileConverter.sh . pdf ps png
```

   - `bin/logReplay.sh`

```
bin/logReplay.sh \
 -i examples/userguide/ch5--trace-monitoring-aspectj/testdata/
      tpmon-20100830-082225522-UTC/ \
 -k true -r false
```

```
diff -r examples/userguide/ch5--trace-monitoring-aspectj/testdata/
    tpmon-20100830-082225522-UTC/tpmon-20100830-082225582-UTC-Thread-2.dat \
<replayed dat file >
```

- `bin/mappingGenerator.sh` — *Not really used so far*

b) Execute user guide examples:

- `ch2-bookstore-application`

```
ant run-jar
```

- `ch2-manual-instrumentation` – copy jars first

```
ant run-monitoring
ant run-analysis -Danalysis.directory=<DIR>
```

- `ch3-4-custom-components` – copy jars first

```
ant run
```

- `ch5-trace-monitoring-aspectj` – copy jars first

```
ant run
```

- `appendix-JMS` – see instructions in user guide
- `JavaEEServletContainerExample` – see instructions in user guide

c) Check user guide PDF `doc/kieker-userguide.pdf`

9. Copy `dist/` to SVN tag and add to repository
10. Merge the `tag/`'s changes (if such) back to the `trunk/`
11. **Publish release files**

    `https://sourceforge.net/apps/trac/sourceforge/wiki/Releasefilesfordownload#Createoreditarelease`

    Uploads via *Project Admin* → *File Manager*

    a) `/`
        i. `kieker-<version>_binaries.zip|tar.gz`
        ii. `kieker-<version>_sources.zip|tar.gz`
        iii. `kieker-userguide.pdf` (e.g., extracted from zip)

    b) `/examples/`
        i. `kieker-examples-JavaEEServletContainerExample.zip|tar.gz`

    c) `/javadoc/`
        i. `kieker-<version>_javadoc.zip|tar.gz`

12. Upload HTML version of Javadoc API
    Only via scp: `SFUSER,kieker@web.sourceforge.net:/home/groups/k/ki/kieker/htdocs/`
    (An ftp client supporting scp can be used, of course)
    a) Create directory `<version>/doc/api/`
    b) Upload contents of unzipped javadoc

5 Testing

5.1 JUnit Tests

5.2 Integration Tests

6 Documentation

6.1 Javadoc API

6.2 User Guide

6.3 Developer Guide

6.4 Web Site

7 Architectural Details

7.1 Goals

- Low Overhead
- Robustness
- Desgined for continuous operation

7.2 Design Decisions

8 Roadmap