JavaDevelopmentEnvironment

The Java Development Environment for Emacs (JDEE) is an integrated development environment for Emacs. It interfaces Emacs to command-line Java development tools, for instance Oracle's JDK.

At some point, development of JDEE was by all appearances slow. JDEE 2.4.1 was released in May, 2013 to support the latest CEDET version, but included no new features. Bug reports, patches, volunteers are very welcome! There was extensive discussion in April/May 2013 about a next generation JDEE (possible fork). Fortunately, there are signs of development for Java in CEDET. Since mid-2015, the projects is much more active (see this graph on GitHub: https://github.com/jdee-emacs/jdee/graphs/contributors).

Project site: https://github.com/jdee-emacs/jdee

Alternatives

EmacsEclim, which provides Eclipse-like features for Java development.

Download

JDEE is available from MELPA.

As of JDEE version 2.4.1, JDEE works with CEDET that comes with Emacs 24.3 release (and is known to work with Emacs 23.4 or later).

Project repo: https://github.com/jdee-emacs/jdee

Bugs

Create an issue at Github: https://github.com/jdee-emacs/jdee/issues

- JDEE does not scale well when opening many Java files. Try ‘emacs `find . -name “*.java”`’ on a mere 50 Java files and you’ll observe each Java file take progressively longer to load. Much more than 50 and you’ll reach intolerable latencies starting Emacs. When I disable JDEE and open the same source code files in stock java-mode, Emacs initializes in a flash (tracked as https://sourceforge.net/p/jdee/bugs/31/)

Features

Core

From the webpage:

- JDEE menu with compile, run, debug, build, browse, project, and help commands
- syntax coloring
- auto indentation
- compile error to source links
- source-level debugging
- source code browsing
- make file support
- ant support
- automatic code generation
- Java source interpreter (Pat Niemeyer’s BeanShell)

JDEE Extensions and Plugins

- Integration with other tools
  - Jdee-Eclipse-integration: different ways to profit from both IDEs or share configuration
  - JdeeAndMavenPom: how to load Maven’s pom.xml
  - JdeeAndMaven2Pom: same, for Maven 2

  - JdeeMaven: still another version for Maven
  - jde-mve seems to be the newest Maven/JDEE integration
  - JdeeGradle: generating Jdee project “prj.el” from Gradle project
  - JdeeDecompile: Integrate with JAD, a decompiler
- Problem searchers
  - JdeeFlymake: on-the-fly syntax checker
  - JdeeFindbugs: looks for bugs in Java code
  - JdeeLint: detects issues with locking, threading, performance, scalability, serialization, …
- Edition tools
  - JdeeJalopy: a Java source code, formatter, beautifier, pretty printer
- Information providers
  - JdeeUsages: finds callers of a method, subclasses, overrides, and other information about your Java classes
- Programming helpers
  - JDIBUG, a new debugger for Java written in Elisp

And more.

JDEE's features explained and demonstrated

- Quick tour through the basic features
- Emacs Java Tutorial: lots of screenshots which show autocompletion, compilation, Ant, code generation, jump to source, wizards, documentation, debugging (even remote debugging of a servlet in Tomcat 5)
- The neverending argument: more about important features, including performance, JSP support, Maven, projects, refactoring.

Others (not so graphical):

- Emacs and JDEE as software development environment: uncomplete,

outdated (from 2003)
- JDEE's user guide (2004)

Link to your success stories here!

- emacs java(jdee) completion note (korean)
- Java Development With Emacs. Using power of Emacs for developing Java applications. Linux, JDEE, Maven, gtags, jdb and a little bit of ELisp together create excellent development environment.

Installation notes

Latest

JDEE is available in MELPA.

As of 2013-06-14, Emacs 24.3 and JDEE 2.4.1 are latest releases. This JDEE release will work with Emacs 23.4 or later. See Download section above to obtain JDEE.

Next all you need in your init file is something like:

```
(add-to-list 'load-path (format "%s/dist/jdee-2.4.1/lisp" my-jdee-path))
(autoload 'jde-mode "jde" "JDE mode" t)
(setq auto-mode-alist
      (append '(("\\.java\\'" . jde-mode)) auto-mode-alist))
```

See http://jdee.sourceforge.net/install.html for more details.

Emacs 24 and older JDEE

Download CEDET 1.1 and load it in Elisp. Emacs comes with CEDET 2.0; you must load CEDET 1.1 before other packages load the version in Emacs. Older JDEE are not compatible with new CEDET.

When downloading the prebuilt older release, you’ll need some Elisp to prevent fatal errors:

```
(setq jde-check-version-flag nil)
(define-obsolete-function-alias 'make-local-hook 'ignore "21.1")
(unless (fboundp 'semantic-format-prototype-tag-java-mode)
  (defalias 'semantic-format-prototype-tag-java-mode 'semantic-format-tag-prot
(require 'hippie-exp)
```

Use something like this to load JDEE for .java files:

```
(add-to-list 'load-path (format "%s/lisp" "~/.emacs.d/jdee-2.4.0.1" "Path to
(autoload 'jde-mode "jde" "JDE mode." t)
(setq auto-mode-alist
      (append '(("\\.java\\'" . jde-mode)) auto-mode-alist))
```

JDEE 2.4.0.1 was confirmed to work in the 24.2.90 pretest release with this approach.

See also: http://forums.fedoraforum.org/showthread.php?t=280711 . Note that editing jde.el and deleting jde.elc is not necessary when setting jde-check-version-flag per above.

Emacs 23

If you are using Emacs 23.4 or later use use the latest release (JDEE 2.4.1) without external dependencies as described above.

Install notes for Gnu Emacs 23 and earlier JDEE releases below. This will help you get (require ‘jde) running. No guarantee on anything else.

jde depends: cedet, elib.

```
(add-to-list 'load-path "/usr/share/emacs/site-lisp/cedet-common/")
```

```
(add-to-list 'load-path "/usr/share/emacs/site-lisp/cedet-contrib/")

(require 'cedet)

(add-to-list 'load-path "/usr/share/emacs/site-lisp/elib/")
```

Then you need to edit jdel.el in lisp dir.

Remove or comment out the following lines, because autoload is now part of emacs, and there is no jde-autoloads.el in jde src.

```
(require 'jde-autoload)
....
(unless
    (and jde-xemacsp
         (file-exists-p
          (expand-file-name
           "jde/auto-autoloads.el"
           (jde-root))))
         (require 'jde-autoload))
```

For jde itself, add these to .emacs, change the dirs to fit your install of course:

```
(add-to-list 'load-path "~/.emacs.d/jde-current/")
(require 'jde)
(setq jde-web-browser "firefox")
(setq jde-doc-dir "~/d/jdk-6-doc/")
```

I don’t know much about it myself. I guess there are more problems waiting.

How to load JDE in Emacs >=23.2 (after CEDET integration into Emacs trunk):

Use the latest release, JDEE 2.4.1! For prior JDEE releases:

I fear it is complex, as the CEDET inclusion changed package names but JDEE is still using the old ones (tested in branches 2.4 and trunk). For instance (require ‘semantic-ctxt) is the old name (standalone CEDET), but in the Emacs-included CEDET it should be transformed to (require ‘semantic/ctxt). Someone must update JDEE. 24.m3.2010, DanielClemente

I finally got JDE to work (mostly) with emacs 23.2. Here is what I did:

1) Download trunk source from svn. 2) Extract the source 3) ant configure 4) ant

here is my init file… There a few ‘tricks’ in here:

```
(add-to-list 'load-path (expand-file-name "~/Documents/elisp/jdee/lisp"))

 (setq semantic-default-submodes '(global-semantic-idle-scheduler-mode
                                     global-semanticdb-minor-mode
                                     global-semantic-idle-summary-mode
                                     global-semantic-decoration-mode
                                     global-semantic-highlight-func-mode
                                     global-semantic-stickyfunc-mode
                                     global-semantic-mru-bookmark-mode))

(add-to-list 'load-path (expand-file-name "~/Documents/elisp/jdibug-0.2"))
(setq semantic-load-turn-everything-on t)
(semantic-mode 1)
(require 'semantic/senator)
(require 'semantic)
(require 'semantic/ia)
(require 'semantic/wisent)
(require 'semantic/wisent/java-tags)


(autoload 'wisent-java-default-setup "wisent" "Hook run to setup Semantic in '
(setq jde-auto-parse-enable nil)
(setq jde-enable-senator nil)
(load "jde-autoload")


(require 'jde-testng)


(require 'jde-maven2)


(require 'jdibug)



(setq defer-loading-jde nil)





(if defer-loading-jde
    (progn
      (autoload 'jde-mode "jde" "JDE mode." t)
      (setq auto-mode-alist
            (append
             '(("\\.java\\'" . jde-mode))
             auto-mode-alist)))
  (require 'jde))

(setq
 jde-sourcepath '( "/Users/ldangelo/Development" )
 jde-db-option-connect-socket '(nil "28380")
 jde-jdk-registry (quote (
                          ("1.5" . "/System/Library/Frameworks/JavaVM.framewo
                          ("1.6" . "/System/Library/Frameworks/JavaVM.framewo
                          )
                         )
 jde-jdk `("1.6")

)
```

```
(setq shell-file-name "bash")
(setq shell-command-switch "-c")
(setq explicit-shell-file-name shell-file-name)
(setenv "SHELL" shell-file-name)
(setq explicit-sh-args '("-login" "-i"))
(if (boundp 'w32-quote-process-args)
  (setq w32-quote-process-args ?\"))



(setq my-emacs-dir (concat (getenv "HOME") "/.emacs.d/tmp/emacs-jde"))


(when (locate-library "semantic")
  (let ((semcach (concat my-emacs-dir "/semantic-cache")))
    (unless (file-directory-p semcach)
      (make-directory semcach))
    (setq semanticdb-default-save-directory semcach)))

(define-key jde-mode-map [f8]   'jdibug-step-over)
(define-key jde-mode-map [M-f8] 'jdibug-step-into)
(define-key jde-mode-map [f7]   'jdibug-step-out)
(define-key jde-mode-map [M-f7] 'jdibug-resume)

(require 'flymake)

(defun skip-cleanup())



(defun semantic-parse())

(defun flymake-java-ecj-init ()
  (let* ((temp-file   (flymake-init-create-temp-buffer-copy
                       'jde-ecj-create-temp-file))
         (local-file  (file-relative-name
                       temp-file
```

```
                         (file-name-directory buffer-file-name))))

        (list "java" (list "-jar" "/Users/ldangelo/Development/ecj.jar" "-Xemacs"
                         "-source" "1.5" "-target" "1.5" "-proceedOnError"
                         "-classpath"
                         (jde-build-classpath jde-global-classpath) local-file))

    (defun flymake-java-ecj-cleanup ()
      "Cleanup after `flymake-java-ecj-init' -- delete temp file and dirs."
      (flymake-safe-delete-file flymake-temp-source-file-name)
      (when flymake-temp-source-file-name
        (flymake-safe-delete-directory (file-name-directory flymake-temp-source-fi

    (defun jde-ecj-create-temp-file (file-name prefix)
      "Create the file FILE-NAME in a unique directory in the temp directory."
      (file-truename (expand-file-name (file-name-nondirectory file-name)
                                       (expand-file-name  (int-to-string (random))

    (push '(".+\\.java$" flymake-java-ecj-init flymake-java-ecj-cleanup) flymake-a

    (push '("\\(.*?\\):\\([0-9]+\\): error: \\(.*?\\)\n" 1 2 nil 2 3 (6 compilatic

    (push '("\\(.*?\\):\\([0-9]+\\): warning: \\(.*?\\)\n" 1 2 nil 1 3 (6 compilat



    (defun my-jde-mode-hook ()
      "Hook for running java file..."
      (message " Loading my-jde-mode-hook...")
      (c-set-offset 'substatement-open 0)
      (c-set-offset 'statement-case-open 0)
      (c-set-offset 'case-label '+)
     (wisent-java-default-setup)
     (flymake-mode)
      (setq
       indent-tabs-mode nil
       tab-width 4
       c-basic-offset 2
       tempo-interactive t
       ))

    (add-hook 'jde-mode-hook 'my-jde-mode-hook)
```

Distributions

- Debian Wheezy: JDEE 2.4.1 works with included Emacs 23.4
- Debian Etch - installed and loaded correctly
- Ubuntu 7.10 (Gutsy Gibbon) - requires a patch JDE Ubuntu Patch

Note from 2004:

If you use the Debian Sid `jde` package with CVS Emacs, you may need to apply this patch.

JDEE compared with other IDEs

- JdeeVsEclipse. Still to write

Frequently Given Answers (FGA)

Did you install it correctly as documented? The website says: “Nearly all the JDEE problems that I have seen are caused by faulty setups,” and “Most problems reported by users are installation/setup problems.” See the Trouble Shooting Guide (Dead link).

Alternatives

malabar-mode

malabar-mode is an effort to create a better Java mode for Emacs.

Eclim

Eclim uses Eclipse as a backend to provide intelligent Java completion and coding support in Emacs

Comments

moved to the Talk page

CategoryProgrammerUtils CategoryModes CategoryProject CategoryJava