


Clojure Core

[Quick Ref](#)
[Short Descs](#)
[Vars Only](#)
[Alphabetical](#)

Namespaces

[clojure](#)
[core](#)
[protocols](#)
[inspector](#)
[java](#)
[browse](#)
[io](#)
[javadoc](#)
[shell](#)
[main](#)
[pprint](#)
[repl](#)
[set](#)
[stacktrace](#)
[string](#)
[template](#)
[test](#)
[junit](#)
[tap](#)
[walk](#)
[xml](#)
[zip](#)

You're viewing version 1.2.0 of load-file. The latest stable version of Clojure Core is 1.3.0.

[doc](#) [examples](#) [comments](#)1.2.0 | 

clojure.core

load-file

(load-file name)

Sequentially read and evaluate the set of forms contained in the file.

© Rich Hickey. All rights reserved. Eclipse Public License 1.0

3 Examples

[top](#)[link](#) | [changes](#)

```
1 ;; Very useful from a REPL
2 ;; Paths are specified as strings using canonical file path notation
3 ;; (rather than clojure-style namespaces dependent on the JVM classpath).
4 ;; The working directory is set to wherever you invoked the JVM from,
5 ;; likely the project root.
6
7 (load-file "src/mylib/core.clj")
8
9 ;; now you can go and evaluate vars defined in that file.
```

[link](#) | [changes](#)

```
1 ;; file located at src/address_book/core.clj
2 ;; current dir is src/..
3
4 (load-file "src/address_book/core.clj")
```

[link](#) | [changes](#)

```
01 ;; create a clojure file on the fly using spit
02 ;; then load it into the REPL and use its function
03
04 user=> (spit "mycode.clj" "(defn doub [x] (* x 2))")
05 nil
06 user=> (load-file "mycode.clj")
07 #'user/doub
08 user=> (doub 23)
09 46
10 user=>
```

Log in to add / edit an example.

See Also

[top](#)[clojure.core/load](#)

0

Loads Clojure code from resources in classpath. A path is interpreted

[clojure.core/spit](#)

0

Opposite of slurp. Opens f with writer, writes content, then close

Log in to add a see also.

Comments

[top](#)

No comments for load-file. Log in to add a comment.