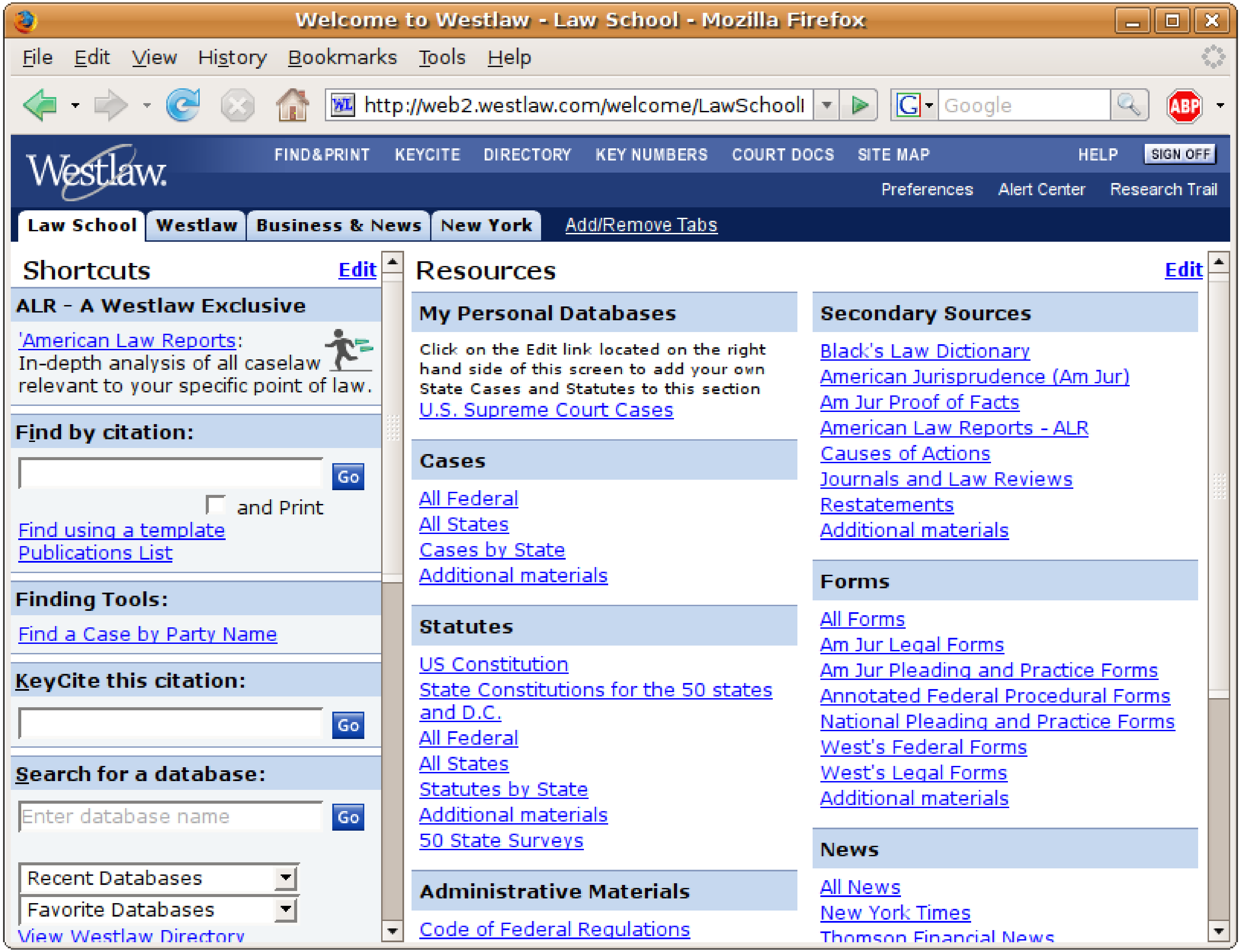
AltLaw and Clojure

Reston, VA
May 21, 2009

Stuart Sierra


Welcome to Westlaw - Law School - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://web2.westlaw.com/welcome/LawSchool
Google
FIND&PRINT KEYCITE DIRECTORY KEY NUMBERS COURT DOCS SITE MAP HELP SIGN OFF
Preferences Alert Center Research Trail
Law School Westlaw Business & News New York Add/Remove Tabs

Shortcuts Edit

ALR - A Westlaw Exclusive

'American Law Reports: In-depth analysis of all caselaw relevant to your specific point of law.

Find by citation:

Go

and Print

Find using a template
Publications List

Finding Tools:

Find a Case by Party Name

KeyCite this citation:

Go

Search for a database:

Enter database name Go

Recent Databases
Favorite Databases
View Westlaw Directory

Resources Edit

My Personal Databases

Click on the Edit link located on the right hand side of this screen to add your own State Cases and Statutes to this section
U.S. Supreme Court Cases

Cases

All Federal
All States
Cases by State
Additional materials

Statutes

US Constitution
State Constitutions for the 50 states and D.C.
All Federal
All States
Statutes by State
Additional materials
50 State Surveys

Administrative Materials

Code of Federal Regulations

Secondary Sources

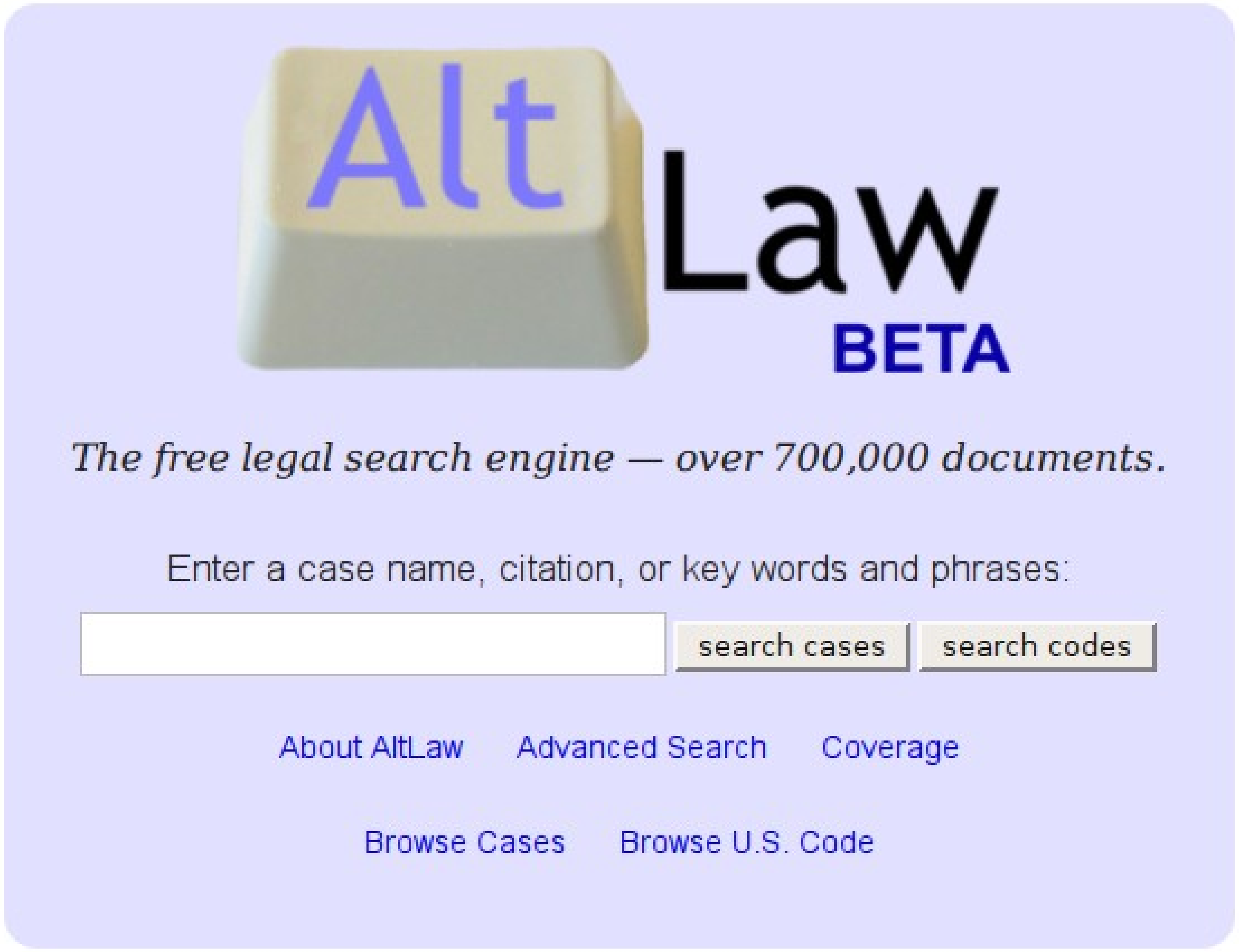
Black's Law Dictionary
American Jurisprudence (Am Jur)
Am Jur Proof of Facts
American Law Reports - ALR
Causes of Actions
Journals and Law Reviews
Restatements
Additional materials

Forms

All Forms
Am Jur Legal Forms
Am Jur Pleading and Practice Forms
Annotated Federal Procedural Forms
National Pleading and Practice Forms
West's Federal Forms
West's Legal Forms
Additional materials

News

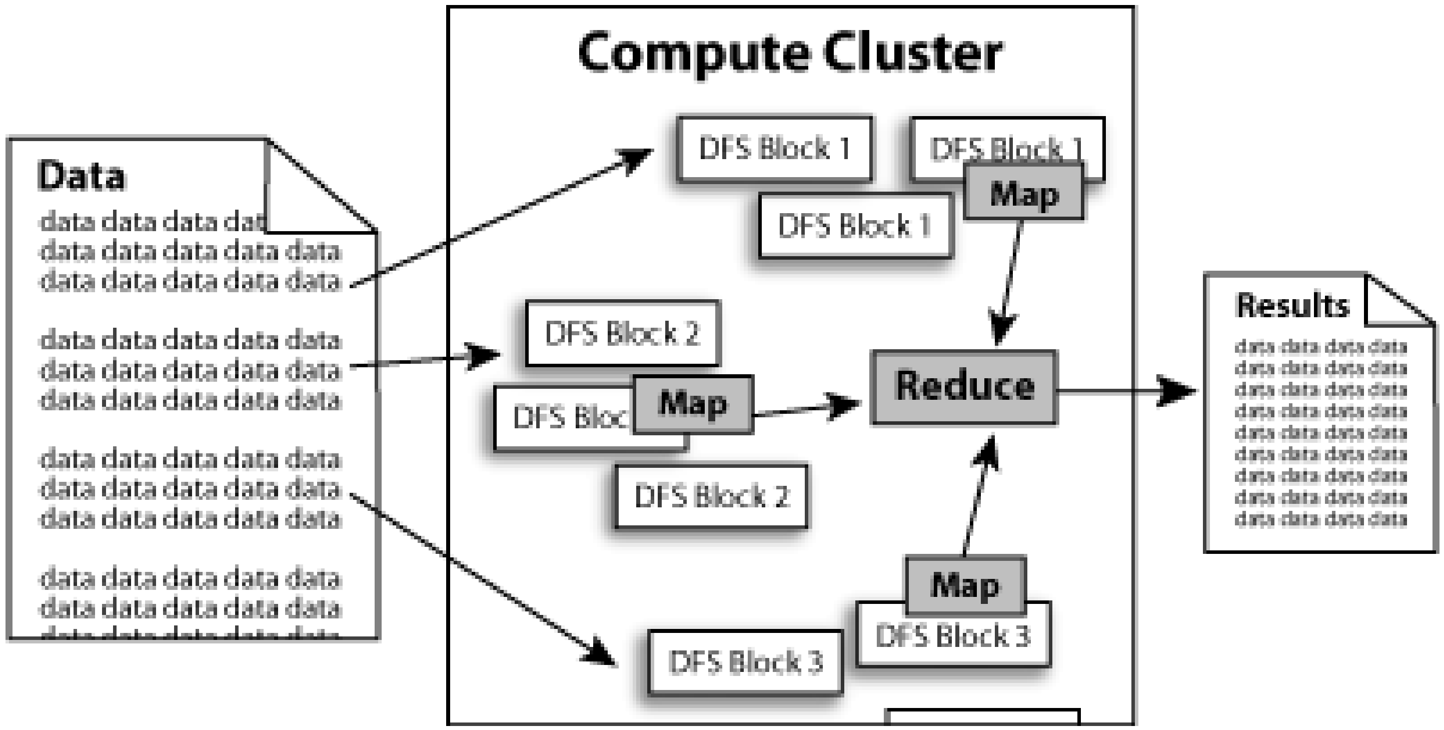
All News
New York Times
Thomson Financial News



The free legal search engine — over 700,000 documents.

Enter a case name, citation, or key words and phrases:

hadoop


Compute Cluster
Data
data data data data data
DFS Block 1
DFS Block 1
DFS Block 1
Map
DFS Block 2
DFS Bloc
Map
DFS Block 2
Reduce
Map
DFS Block 3
DFS Block 3
Results
data data data data

```
public static class MapClass extends MapReduceBase
  implements Mapper<LongWritable, Text, Text, IntWritable> {

  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(LongWritable key, Text value,
                  OutputCollector<Text, IntWritable> output,
                  Reporter reporter) throws IOException {
    String line = value.toString();
    StringTokenizer itr = new StringTokenizer(line);
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      output.collect(word, one);
    }
  }
}
```

```
public static class Reduce extends MapReduceBase
  implements Reducer<Text, IntWritable, Text, IntWritable> {

  public void reduce(Text key, Iterator<IntWritable> values,
                     OutputCollector<Text, IntWritable> output,
                     Reporter reporter) throws IOException {
    int sum = 0;
    while (values.hasNext()) {
      sum += values.next().get();
    }
    output.collect(key, new IntWritable(sum));
  }
}
```

```
(map key value)

(reduce key values)
```

```
(setup-mapreduce)

(defn my-map [key value]
 ... return list of [key,value] pairs)

(defn my-reduce [key values]
 ... return list of [key,value] pairs)
```

Apache Solr

Lucene

```
(ns org.altlaw.www.DocResource
  (:gen-class :extends org.restlet.resource.Resource))

(defn -getVariants [this]
  ... return list of supported media types ...)

(defn -represent [this variant]
  ... respond to GET request ...)

(defn -acceptRepresentation [this]
  ... respond to POST request ...)

(defn -storeRepresentation [this entity]
  ... respond to PUT request ...)

(defn -deleteRepresentation [this]
  ... respond to DELETE request ...)
```

StringTemplate

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>$html_title$</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <meta http-equiv="Content-Style-Type" content="text/css" />
    $html_head$
  </head>
  <body>
    $html_body$
  </body>
</html>
```

StringTemplate

```
$xhtml_page(
    html_head=default_html_head(),
    html_body={
        <div id="container" class="$page_class$">
          <div id="site_head">$site_head$</div>
          <div id="site_body">$site_body$<div id="content">
          <div id="site_foot">$site_foot$</div>
        </div>
        $analytics()$
    }
)$
```

test-is assertions

```
(is (= 4 (+ 2 2)))
true

(is (= 5 (+ 2 2)))
FAIL in ...
expected: (= 5 (+ 2 2))
  actual: (not (= 5 4))

(is (instance? Integer (/ 3 5)))
FAIL in ...
expected: (instance? Integer (/ 3 5))
  actual: clojure.lang.Ratio
```

test-is assertions

```
(is (thrown? ArithmeticException (/ 1 0)))
#<ArithmeticException java.lang.ArithmeticException: Divide
by zero>


(is (thrown? IllegalArgumentException (/ 1 0)))
ERROR in ..
expected: (thrown? IllegalArgumentException (/ 1 0))
  actual: java.lang.ArithmeticException: Divide by zero
 at clojure.lang.Numbers.divide (Numbers.java:138)
    user/eval (NO_SOURCE_FILE:1)
    clojure.lang.Compiler.eval (Compiler.java:4580)
    clojure.core/eval (core.clj:1728)
    swank.commands.basic/eval_region (basic.clj:36)
```

Clojure tests as metadata

```
(defn add
  ([x y] (+ x y))
  {:test (fn [] (assert (= 7 (add 3 4))))})


(test #'add)
:ok
```

test-is tests as metadata

```
(with-test
    (defn add [x y] (+ x y))
  (is (= 7 (add 3 4)))
  (is (= 8 (add 2 2))))

(run-tests)
Testing user

FAIL in (add) ...
expected: (= 8 (add 2 2))
  actual: (not (= 8 4))

Ran 1 tests containing 2 assertions.
1 failures, 0 errors.
```

test-is tests in isolation

```
(deftest addition
  (is (= 4 (add 2 2)))
  (is (= 7 (add 3 4)))
  (is (= 9 (add 5 5))))
```

```
(addition)
FAIL in (addition) ...
expected: (= 9 (add 5 5))
  actual: (not (= 9 10))
```

assertions with shared structure

```
(deftest addition
  (are (= _1 (add _2 _3))
       4  2 2
       7  3 4
       9  5 5))
```

```
(deftest addition
  (is (= 4 (add 2 2)))
  (is (= 7 (add 3 4)))
  (is (= 9 (add 5 5))))
```

code walker

```
(defn walk [inner outer form]
  (cond
   (list? form) (outer (apply list (map inner form)))
   (seq? form) (outer (doall (map inner form)))
   (vector? form) (outer (vec (map inner form)))
   (map? form) (outer (into (outer (if (sorted? form)
                                      (sorted-map) {}))
                            (map inner form)))
   (set? form) (outer (into (outer (if (sorted? form)
                                      (sorted-set) #{}))
                            (map inner form)))
   :else (outer form)))
```

code walker

Post-order traversal

```
(defn postwalk [f form]
  (walk (partial postwalk f) f form))
```

Pre-order traversal

```
(defn prewalk [f form]
  (walk (partial prewalk f) identity (f form)))
```

using the code walker

```
(defn macroexpand-all [form]
  (prewalk (fn [x]
             (if (seq? x) (macroexpand x) x))
           form))


(defn postwalk-replace [smap form]
  (postwalk (fn [x]
              (if (contains? smap x) (smap x) x))
            form))
```

templates

```
(template (= _1 (add _2 (* x y))))
```

```
(let [HOLE_1282 (* x y)]
  (fn [_1 _2] (= _1 (add _2 HOLE_1282))))
```

```
(do-template (is (= _1 (add _2 _3)))
             4   2 2
             7   3 4
             9   5 5))
```

```
(do (is (= 4 (add 2 2)))
    (is (= 7 (add 3 4)))
    (is (= 9 (add 5 5))))
```

assertion templates

```
(deftest addition
  (are (= _1 (add _2 _3))
       4  2 2
       7  3 4
       9  5 5))
```

```
(deftest addition
  (is (= 4 (add 2 2)))
  (is (= 7 (add 3 4)))
  (is (= 9 (add 5 5))))
```

singletons

```
(def *thing* (ThingFactory/getInstance))
```

```
(defn make-new-thing []
  (ThingFactory/getInstance))

(declare *thing*)

(defmacro with-thing [& body]
  (binding [*thing* (make-new-thing)]
    ~@body))
```

global singletons

```
(defn global-singleton [f]
  (let [instance (atom nil)
        make-instance (fn [_] (f))]
    (fn [] (or @instance
               (swap! instance make-instance)))))
```

```
(def thing (global-singleton
             (fn [] (ThingFactory/getInstance))))

(thing)
```

per-thread singletons

```
(defn per-thread-singleton [f]
  (let [thread-local (proxy [ThreadLocal] []
                       (initialValue [] (f)))]
    (fn [] (.get thread-local))))
```

```
(def thing (per-thread-singleton
             (fn [] (ThingFactory/getInstance))))

(thing)
```

altlaw.org
columbialawtech.org

Alt Law

clojure.org
code.google.com/p/clojure-contrib

stuartsierra.com