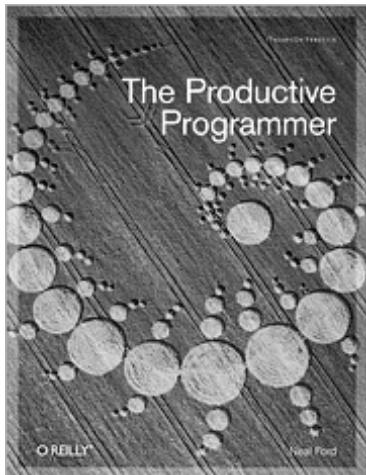


the productive programmer: mechanics



NEAL FORD software architect / meme wrangler

ThoughtWorks

nford@thoughtworks.com
3003 Summit Boulevard, Atlanta, GA 30319
www.nealford.com
www.thoughtworks.com
memeagora.blogspot.com

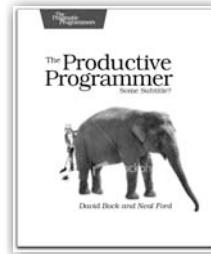
memeagora.blogspot.com
www.memeagora.com
www.memeagora.com

where did this topic come from?

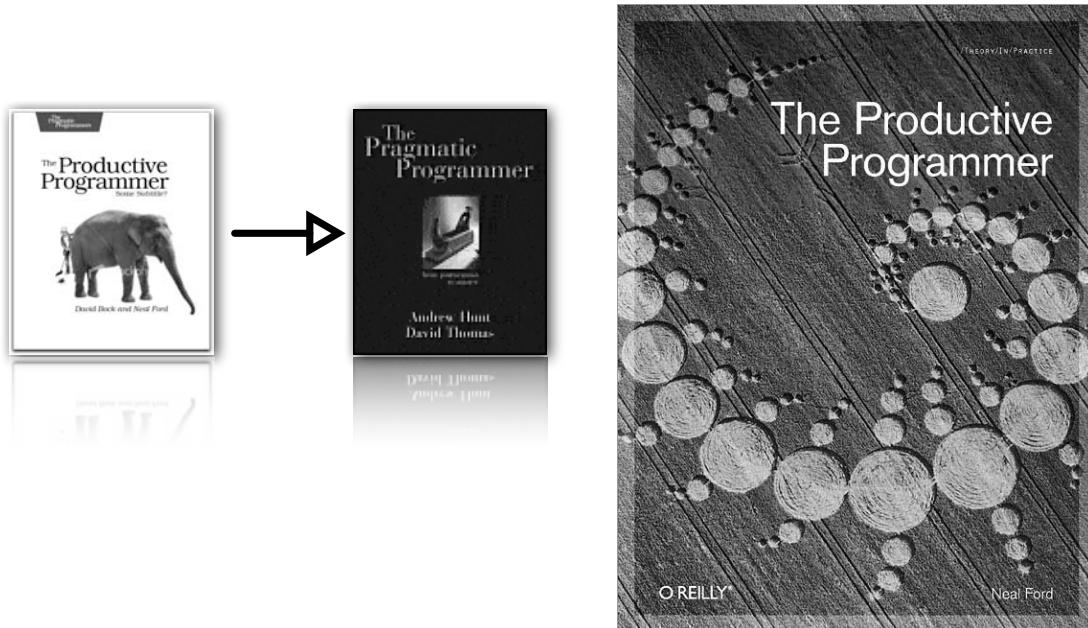


where did this topic come from?

```
[nealford] ~] => ls
2print
Applications
Archive
Desktop
Documents
downloads
images
Library
Mingle
Movies
Music
Neal_Ford-LOP_Shifting_Paradigms-slides.pdf
Notes702Mac.dmg
PartitionMagic805AllWin_English.ZIP
Pictures
Public
Sites
[nealford] ~] => cd ~/work/nealford/
[nealford] ~/work/nealford ]=> svn st | grep '^?*' | tr '?*' '-' | sed 's/[ -]*// ' | sed 's/[ -]/ /g' | xargs svn add
```



where did this topic come from?



what i cover:

acceleration

doing stuff faster

focus

killing distractions

automation

getting your computer to work harder

canonicality

applying the **dry** principle

acceleration



typing is faster than navigation



o/s accelerators

windows explorer address bar (**alt-d**)

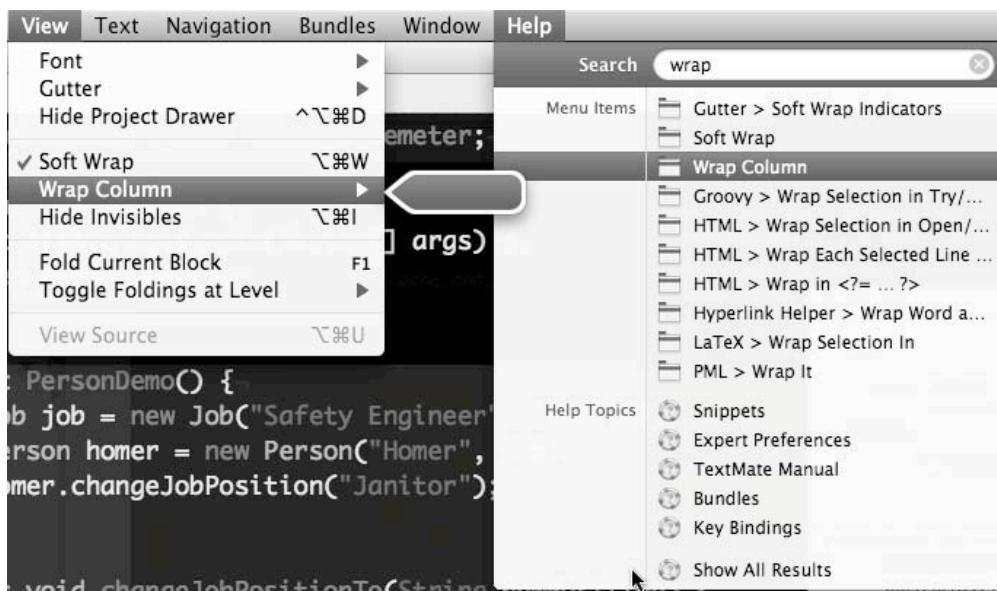
finder (**apple-shift-g**)

firefox

/ searching

number-fox plugin

leopard smart help



clipboards

why do operating systems have only 1 clipboard with 1 entry????

clcl



iClip (\$\$)



jump cut

mac os x : jumpcut

context switching eats time



there and back

pushd pushes a directory on the stack

popd pops it back off

```
C:\temp>pushd \MyDocuments\Documents\dev\haskell
C:\MyDocuments\Documents\dev\haskell>dir
02/13/2006  12:12 AM    <DIR>      .
02/13/2006  12:12 AM    <DIR>      ..
02/13/2006  12:12 AM    <DIR>      nfjs_functionallangs_haskell
          0 File(s)            0 bytes
          3 Dir(s)  11,743,178,752 bytes free

C:\MyDocuments\Documents\dev\haskell>popd
```

pushd/popd

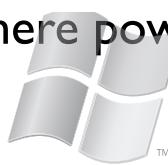


command prompts

graphical explorers better for some things....

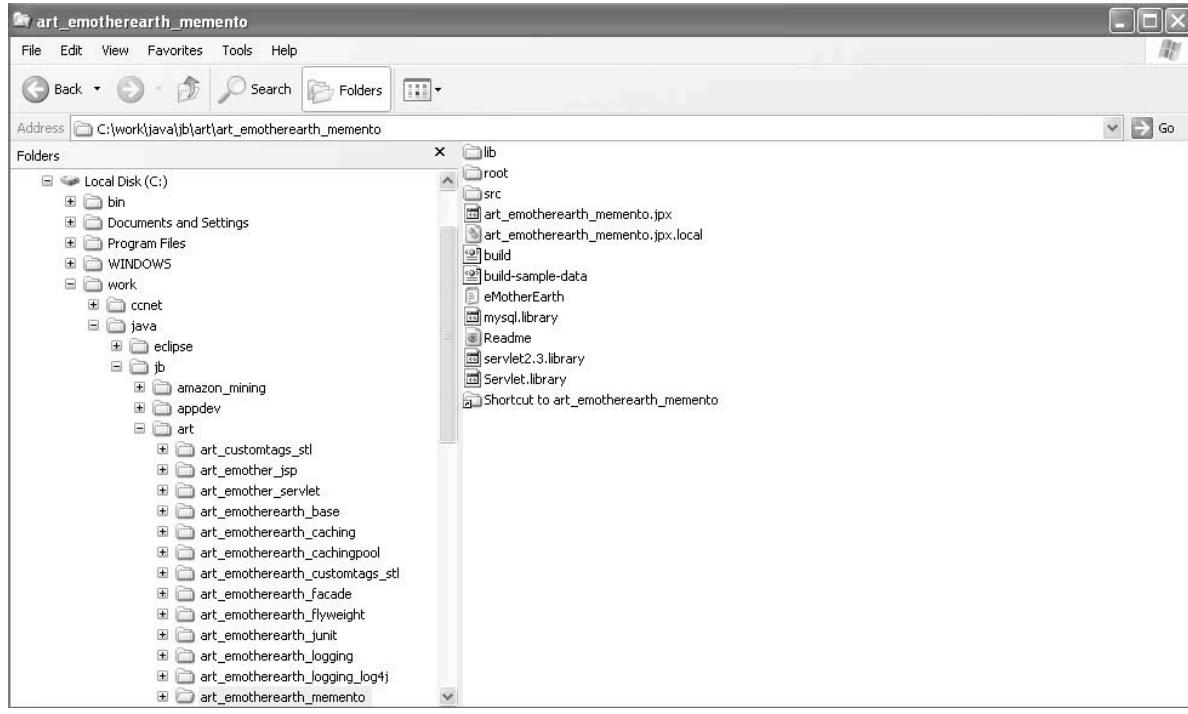
...command line better for others

command prompt here power toy

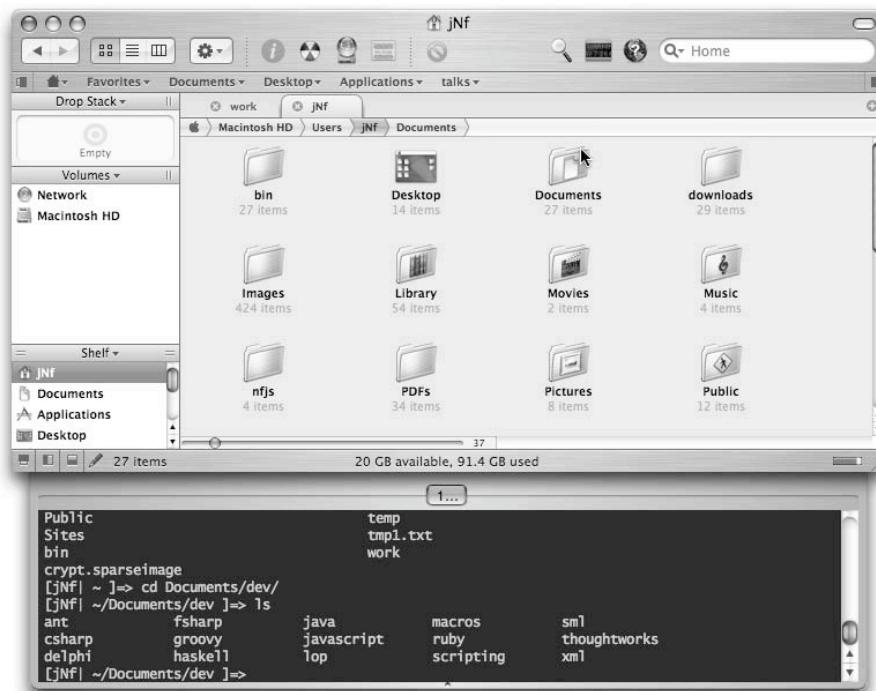
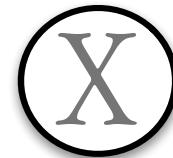


bash here (cygwin)

cmd prompt explorer bar



path finder



how many of you have
written an application for
heads-down data entry
personnel?

when coding, always prefer
keyboard to mouse



learning shortcuts

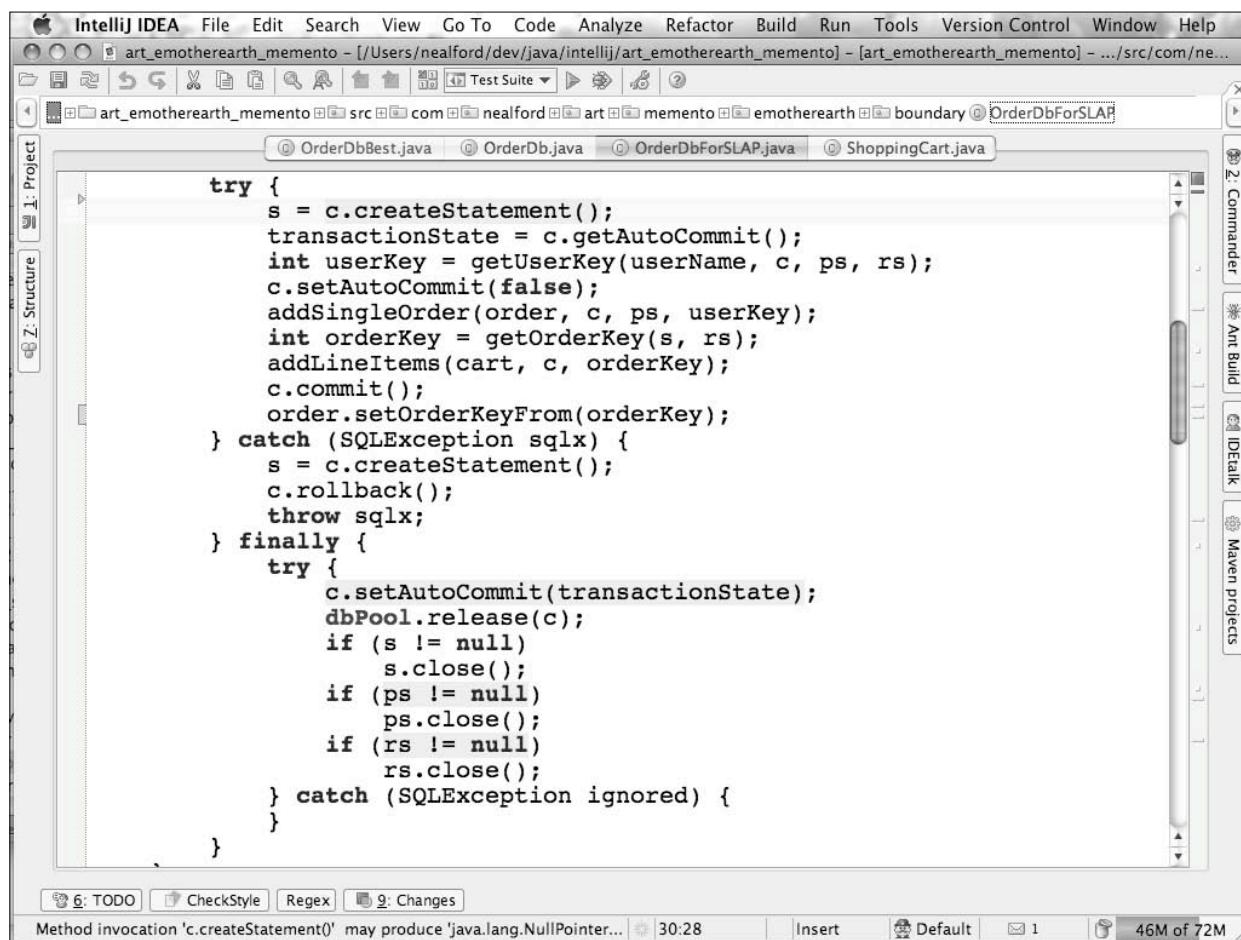
make yourself use the shortcut even if you've gotten there another way

have someone/something pester you about it

pair programmer

key promoter plug-in for intellij

mousefeed for eclipse



The screenshot shows the IntelliJ IDEA interface with the following details:

- Menu Bar:** IntelliJ IDEA, File, Edit, Search, View, Go To, Code, Analyze, Refactor, Build, Run, Tools, Version Control, Window, Help.
- Toolbar:** Standard icons for file operations like Open, Save, Cut, Copy, Paste, Find, etc.
- Project Bar:** art_emotherearth_memento - [/Users/nealford/dev/java/intellij/art_emotherearth_memento] - [art_emotherearth_memento] - .../src/com/ne...
- Editor:** Shows Java code for `OrderDbForSLAP`. The code handles database transactions, statement creation, and resource cleanup.

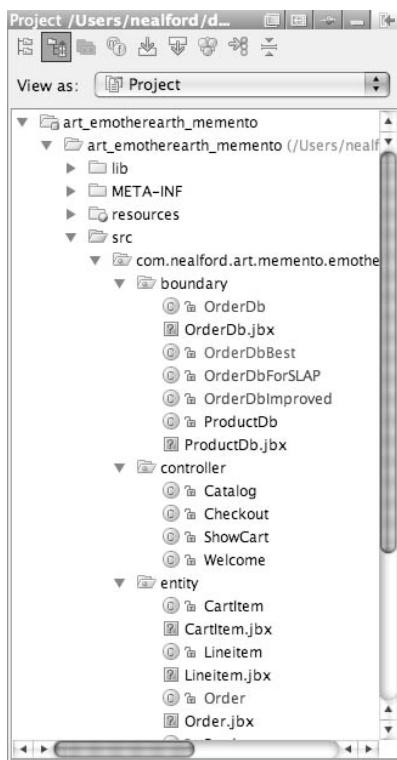
```
try {
    s = c.createStatement();
    transactionState = c.getAutoCommit();
    int userKey = getUserKey(userName, c, ps, rs);
    c.setAutoCommit(false);
    addSingleOrder(order, c, ps, userKey);
    int orderKey = getOrderKey(s, rs);
    addLineItems(cart, c, orderKey);
    c.commit();
    order.setOrderKeyFrom(orderKey);
} catch (SQLException sqlx) {
    s = c.createStatement();
    c.rollback();
    throw sqlx;
} finally {
    try {
        c.setAutoCommit(transactionState);
        dbPool.release(c);
        if (s != null)
            s.close();
        if (ps != null)
            ps.close();
        if (rs != null)
            rs.close();
    } catch (SQLException ignored) {
    }
}
```

- Sidebar:** Includes tabs for `OrderDbBest.java`, `OrderDb.java`, `OrderDbForSLAP.java`, and `ShoppingCart.java`.
- Right Panel:** Shows various tool icons: Commander, Ant Build, IDETalk, and Maven projects.
- Bottom Status Bar:** Shows TODO count (6), CheckStyle, Regex, Changes, and a message about a potential NullPointerException.
- Bottom Footer:** Method invocation information, current time (30:28), Insert mode, Default settings, and page numbers (46M of 72M).

learning shortcuts

repeat them to yourself

flash cards



all our
hierarchies
are too deep:

file system
packages

```
private static final String TO_RETRIEVE_USER_KEY =
    "SELECT ID FROM USERS WHERE NAME = ?";
private static final String SQL_INSERT_LINEITEM =
    "INSERT INTO LINEITEMS (ORDER_KEY, ITEM_ID, QUANTITY) " +
    "VALUES(?, ?, ?)";
private static final String SQL_INSERT_ORDER =
    "INSERT INTO ORDERS (USER_KEY, CC_TYPE, CC_NUM, CC_EXP)" +
    "VALUES (?, ?, ?, ?)";
private DBPool dbPool;

public void addOrderFrom(ShoppingCart cart, String userName,
                        Order order) throws SQLException {
    Map db = setupDataInfrastructure();
    try {
        int userKey = userKeyBasedOn(userName, db);
        add(order, userKey, db);
        addLineItemsFrom(cart,
                         order.getOrderKey(), db);
        completeTransaction(db);
    } catch (SQLException sqlx) {
        rollbackTransactionFor(db);
        throw sqlx;
    } finally {
        cleanUp(db);
    }
}
```

```
private static final String TO_RETRIEVE_USER_KEY =
    "SELECT ID FROM USERS WHERE NAME = ?";
private static final String SQL_INSERT_LINEITEM =
    "INSERT INTO LINEITEMS (ORDER_KEY, ITEM_ID, QUANTITY) " +
    "VALUES(?, ?, ?)";
private static final String SQL_INSERT_ORDER =
    "INSERT INTO ORDERS (USER_KEY, CC_TYPE, CC_NUM, CC_EXP)" +
    "VALUES (?, ?, ?, ?)";
private DBPool dbPool;

public void addOrderFrom(ShoppingCart cart, String userName,
                        Order order) throws SQLException {
    Map db = setupDataInfrastructure();
    try {
        int userKey = userKeyBasedOn(userName, db);
        add(order, userKey, db);
        addLineItemsFrom(cart,
                         order.getOrderKey(), db);
        completeTransaction(db);
    } catch (SQLException sqlx) {
        rollbackTransactionFor(db);
        throw sqlx;
    } finally {
        cleanUp(db);
    }
}
```

A screenshot of the IntelliJ IDEA IDE interface. The main window shows the code for `ShoppingCart.java`. A modal dialog box with a dark gray background and white text is centered over the code, displaying the text "introduce variable". The code itself is as follows:

```
public void restoreFromBookmark(ShoppingCartMemento memento) {
    this.itemList = memento.restoreMemento();
}

public class ShoppingCartMemento {
    private List itemList;

    public List restoreMemento() {
        return itemList;
    }

    public void saveMemento() {
        List mementoList = ShoppingCart.this.itemList;
        itemList = new ArrayList(mementoList.size());
        Iterator i = mementoList.iterator();
        while (i.hasNext())
            itemList.add(i.next());
    }
}
```

The IntelliJ interface includes toolbars, a project tree on the left, and various tabs and status bars at the bottom.

A screenshot of the IntelliJ IDEA IDE interface, similar to the one above, showing the same code for `ShoppingCart.java`. A modal dialog box with a dark gray background and white text is centered over the code, displaying the text "introduce variable redux". The code is identical to the first screenshot.

```
public void restoreFromBookmark(ShoppingCartMemento memento) {
    this.itemList = memento.restoreMemento();
}

public class ShoppingCartMemento {
    private List itemList;

    public List restoreMemento() {
        return itemList;
    }

    public void saveMemento() {
        List mementoList = ShoppingCart.this.itemList;
        itemList = new ArrayList(mementoList.size());
        Iterator i = mementoList.iterator();
        while (i.hasNext())
            itemList.add(i.next());
    }
}
```

The IntelliJ interface includes toolbars, a project tree on the left, and various tabs and status bars at the bottom.

A screenshot of the IntelliJ IDEA IDE interface. The main window shows Java code for the `OrderDb` class. A modal dialog box is overlaid on the code area, containing the text "escalating selection". The code itself is a method for adding an order to a shopping cart, involving database connections and statements.

```
public void addOrder(final ShoppingCart cart, String userName,
                     Order order) throws SQLException {
    connection = null;
    stmt = null;
    try {
        connection = dbPool.getConnection();
        insertOrder(getUserKey(userName), order);
        int orderKey = getGeneratedOrderKey();
        insertLineItems(cart, orderKey);
        commitOrder(order, orderKey);
    } catch (SQLException sqlx) {
        connection.rollback();
        throw sqlx;
    } finally {
        try {
            dbPool.release(connection);
            if (stmt != null) {
                stmt.close();
            }
        } catch (SQLException ignored) {
        }
    }
}

private void insertLineItems(final ShoppingCart cart, int orderKey) th...
```

A screenshot of the IntelliJ IDEA IDE interface, similar to the one above but with a different modal dialog. This dialog contains the text "goto symbol". The code shown is the same as in the first screenshot, a method for adding an order to a shopping cart.

```
public void addOrder(final ShoppingCart cart, String userName,
                     Order order) throws SQLException {
    connection = null;
    stmt = null;
    try {
        connection = dbPool.getConnection();
        insertOrder(getUserKey(userName), order);
        int orderKey = getGeneratedOrderKey();
        insertLineItems(cart, orderKey);
        commitOrder(order, orderKey);
    } catch (SQLException sqlx) {
        connection.rollback();
        throw sqlx;
    } finally {
        try {
            dbPool.release(connection);
            if (stmt != null) {
                stmt.close();
            }
        } catch (SQLException ignored) {
        }
    }
}

private void insertLineItems(final ShoppingCart cart, int orderKey) th...
```

some choice shortcuts

	intelliJ	eclipse
goto class	ctrl-n	ctrl-shift-t
introduce variable	ctrl-alt-v	alt-shift-l
escalating selection	ctrl-w	alt-shift-up
recently edited files	ctrl-e	n/a (ctrl-e)
symbol list	alt-ctrl-shift-n	ctrl-o
incremental search	alt-f3	ctrl-j

live templates

all major ide's and coding text editors

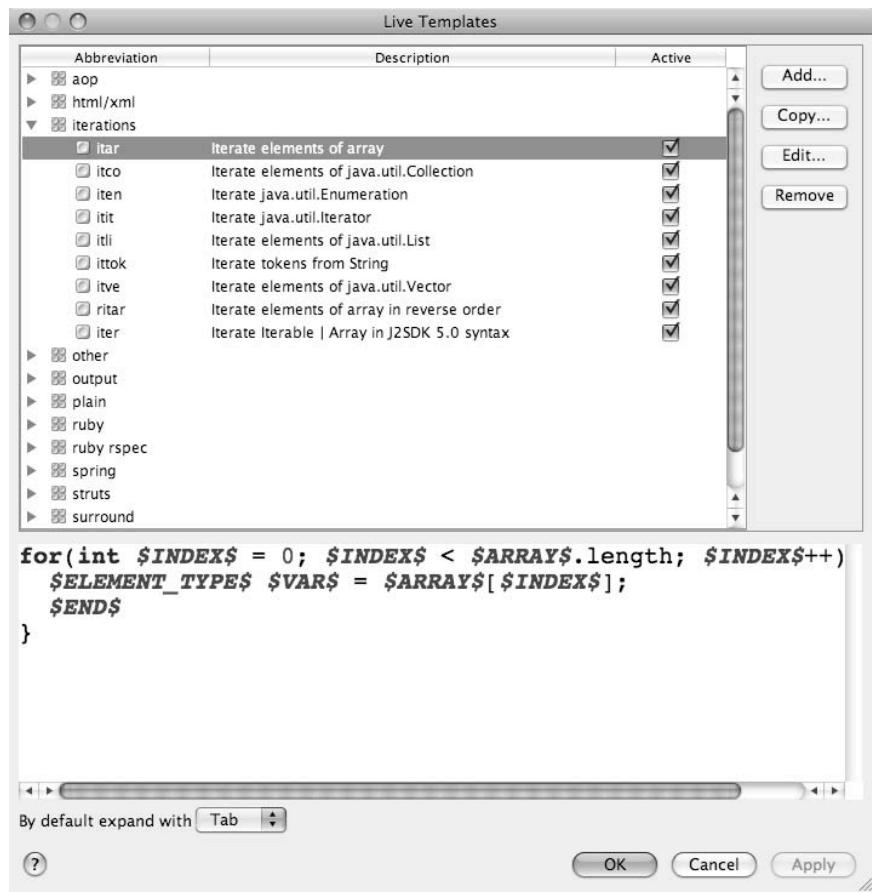
parameter substitution, default values, repeating values

learn the language of your template engine

velocity in intelliJ

bash for textmate/e editor

i t
n e
t m
e P
| l
j a
i t
j e
s



every time you type
something for the 3rd time,
templatize it



key macro tools

live templates at the o/s level

auto-hot key



textexpander



typinator

textexpander

**don't type the same
commands over and over**



simple stuff

get a comfortable chair!

dual monitors...

...sitting immediately in front of you

administrator privilege for the o/s

good keyboard

insidious distractions

modern office environments are terrible for
knowledge workers

too much out of context noise

how many people here work in cube land?

war rooms



locus of attention

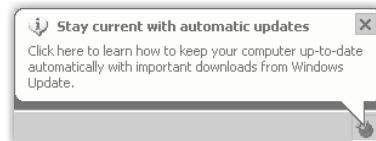
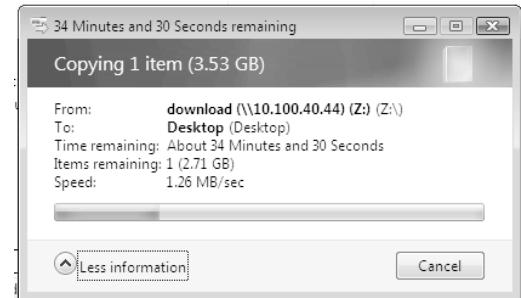
in *the humane interface*, jef raskin describes *locus of attention*

anything that happens outside your locus of attention breaks *flow*

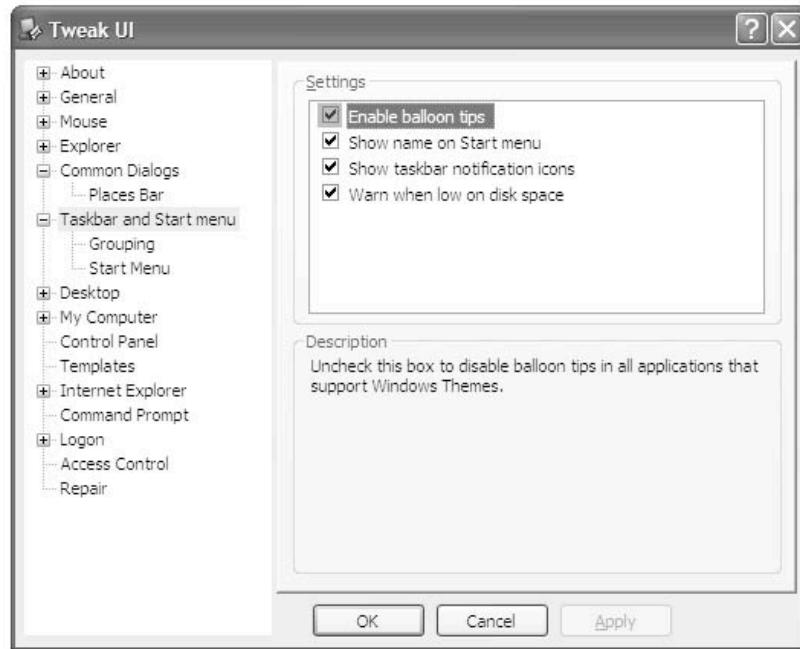
in *flow*, michael csikszentmihalyi describes *flow state*

total concentration

time disappears



killing balloon tips



screen dimmers

automatically makes your background dark
after a set time

jedi concentrate



doodim



the higher the level of
concentration, the denser
the ideas



the easy stuff

turn off notifications

don't keep email open

turn off instant messaging

put on headphones

create office “quiet time”

focus techniques



search > navigation

all developer hierarchies are too deep

file system

package/namespace

documentation

what worked well with 20 mb hard drives fails
with 200 gb

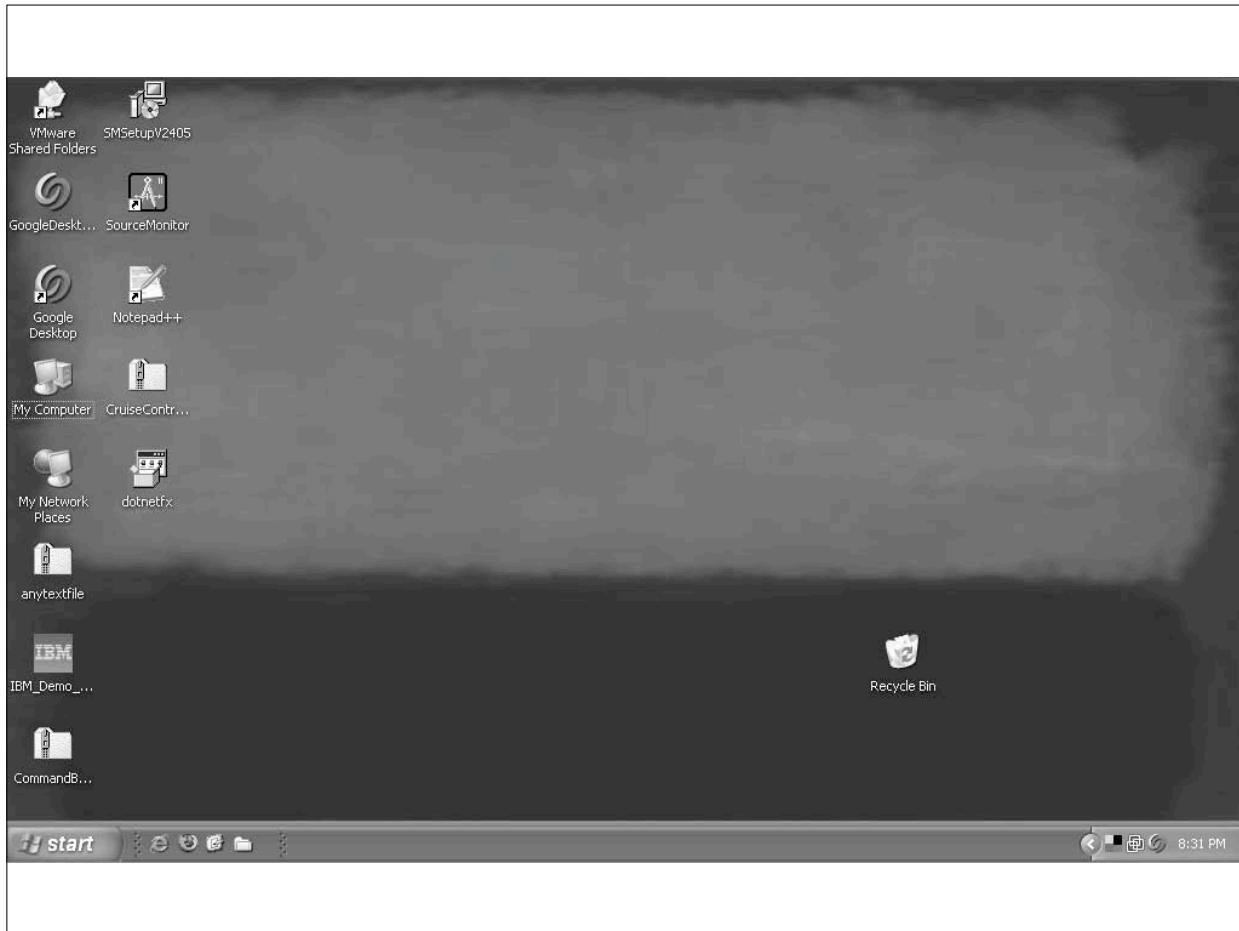
desktop search

built into modern operating systems

retro-fittable in older ones

google desktop search

larry's "any text file" indexer



**replace file hierarchy
navigation with search**



rooted views

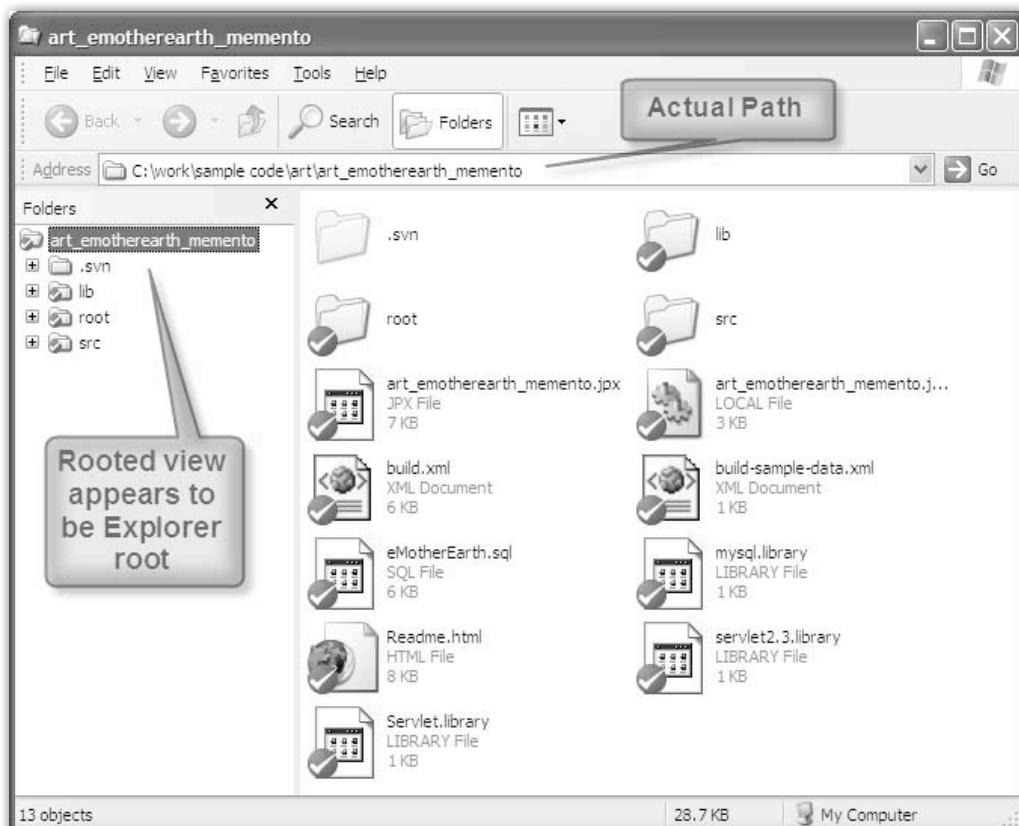
specialized explorer view

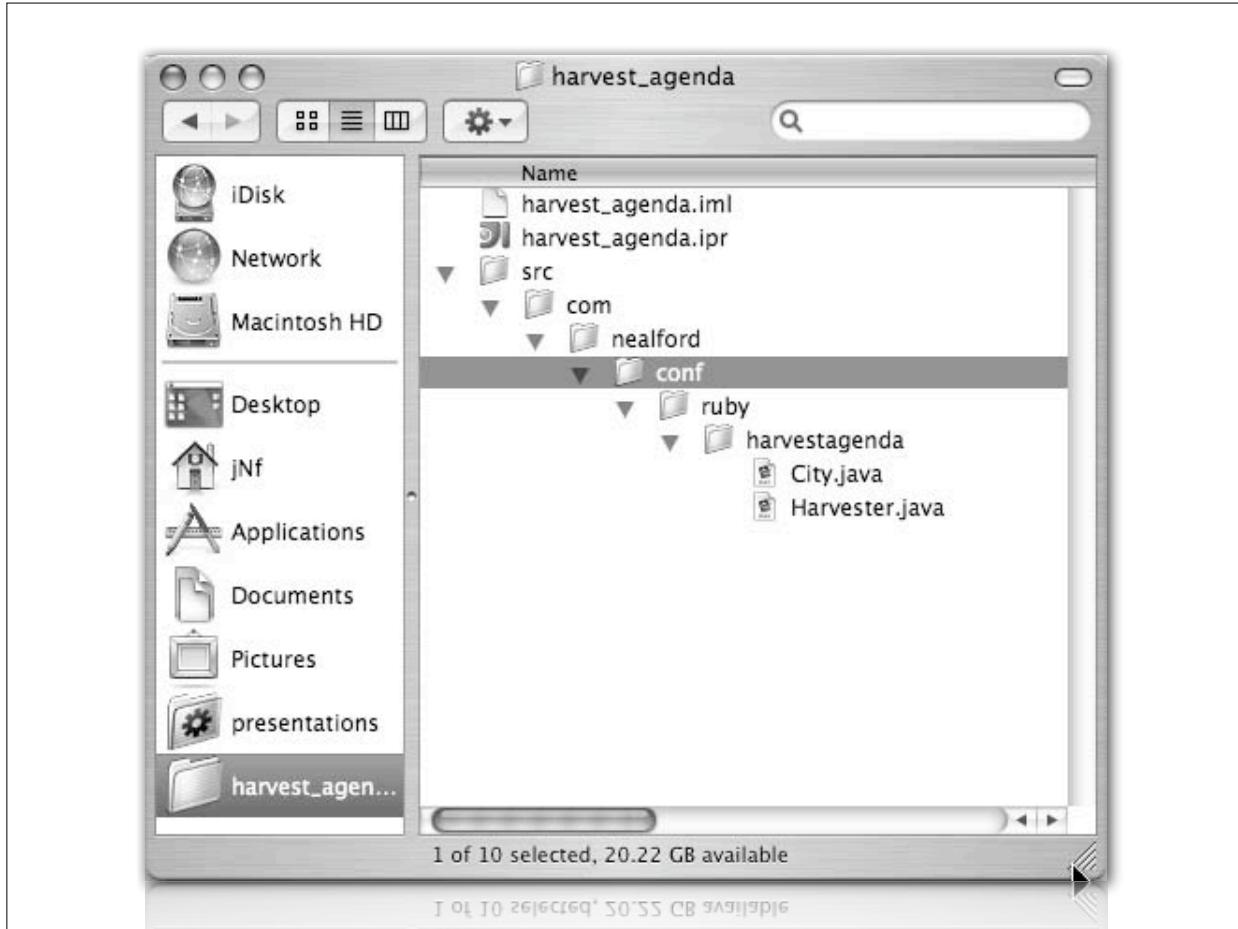
especially good for directory-based version control

rooted view == project explorer

create a shortcut:

```
C:\WINDOWS\explorer.exe /e,/root,c:\work\project
```





use virtual desktops

virtual desktops allow you multiple isolated environments

bind applications to desktops

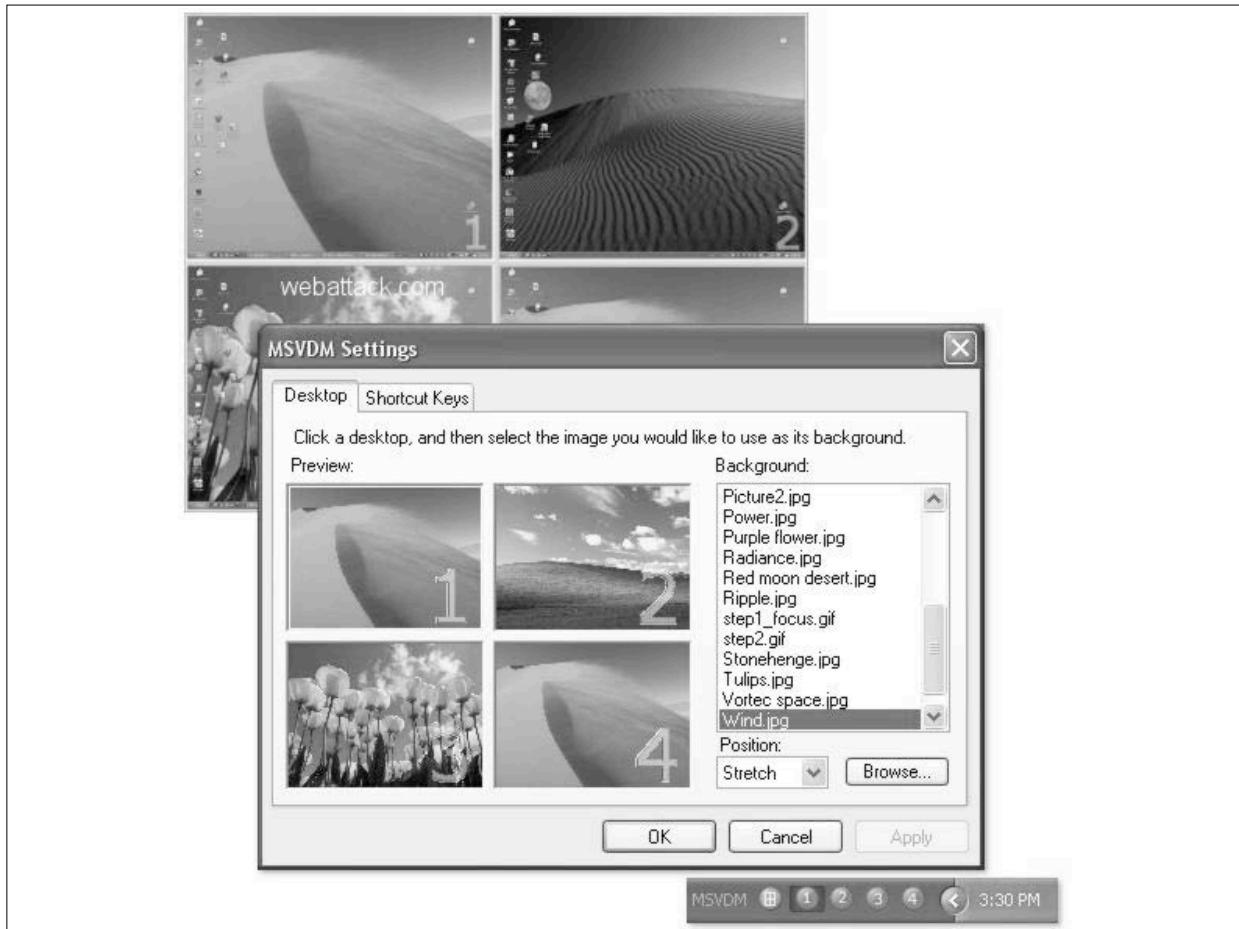
virtual desktop manager power toy

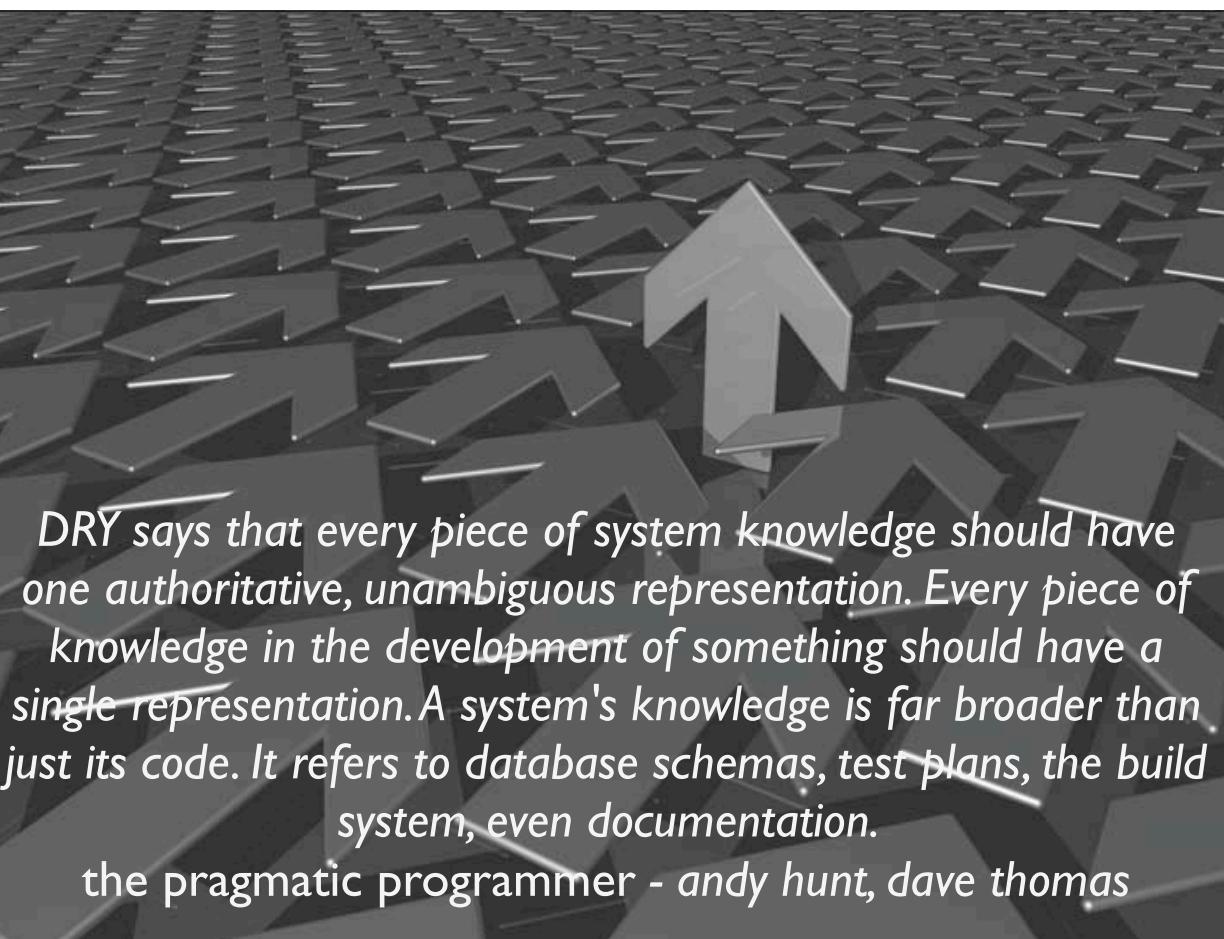


virtue desktops (in tiger)



spaces (in leopard)





DRY says that every piece of system knowledge should have one authoritative, unambiguous representation. Every piece of knowledge in the development of something should have a single representation. A system's knowledge is far broader than just its code. It refers to database schemas, test plans, the build system, even documentation.

the pragmatic programmer - andy hunt, dave thomas

dry o/r

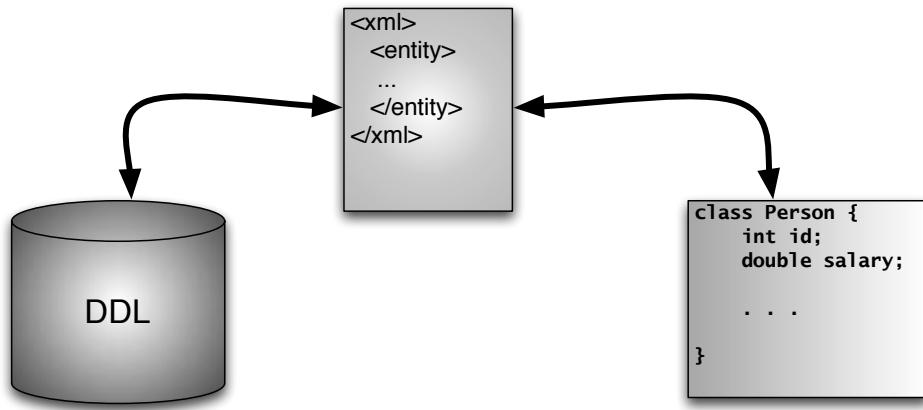
object-relational mapping is one of the most common dry violations

database schema + xml configuration + pojo > I

decide on the canonical representation

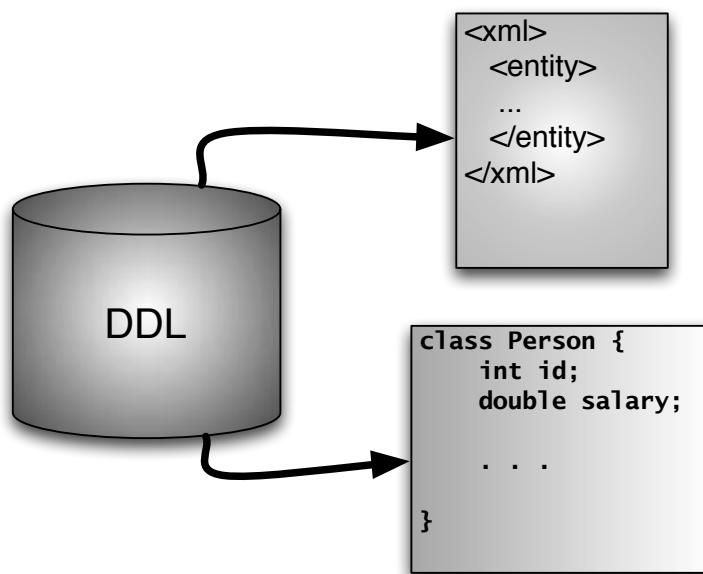
generate the others

the scenario



where's the information?

canonical representation



the target

```
<sqlMap namespace='event'>
    <typeAlias type='com.nealford.conf.canonicality.Event' alias='Event' />
    <resultMap id='eventResult' class='Event'>
        <result property='description' column='DESCRIPTION' />
        <result property='eventKey' column='EVENT_KEY' />
        <result property='start' column='START' />
        <result property='eventType' column='EVENT_TYPE' />
        <result property='duration' column='DURATION' />
    </resultMap>
    <select resultMap='eventResult' id='getEvents'>select * from event where id = ?</select>
    <select resultClass='com.nealford.conf.canonicality.Event' id='getEvent'>
        select * from event where id = #value#
    </select>
</sqlMap>
```

build event sql map

```
class GenerateEventSqlMap {
    static final SQL =
        ["sqlUrl":"jdbc:derby:/Users/jNf/work/derby_data/schedule",
         "driverClass":"org.apache.derby.jdbc.EmbeddedDriver"]
    def _file_name
    def types = [:]

    def GenerateEventSqlMap(file_name) {
        _file_name = file_name
    }
}
```

```

def columnNames() {
    Class.forName(SQL["driverClass"])
    def rs = DriverManager.getConnection(SQL["sqlUrl"]).createStatement().
        executeQuery("select * from event where 1=0")

    def rsmd = rs.getMetaData()
    def columns = []
    for (index in 1..rsmd.getColumnCount()) {
        columns << rsmd.getColumnName(index)
        types.put(camelize(rsmd.getColumnName(index)),
                  rsmd.getColumnTypeName(index))
    }
    return columns
}

```

```

def create_mapping_file() {
    def writer = new StringWriter()
    def xml = new MarkupBuilder(writer)
    xml.sqlMap(namespace:'event') {
        typeAlias(alias:'Event',
                  type:'com.nealford.conf.canonicity.Event')
        resultMap(id:'eventResult', class:'Event') {
            columnMap().each() {key, value ->
                result(property:"${key}", column:"${value}")
            }
        select(id:'getEvents', resultMap:'eventResult',
              'select * from event where id = ?')
        select(id:"getEvent",
              resultClass:"com.nealford.conf.canonicity.Event",
              "select * from event where id = #value#")
    }

    new File(_file_name).withWriter { w ->
        w.writeLine("${writer.toString()}")
    }
}

```

generated sql map

```
<sqlMap namespace='event'>
  <typeAlias type='com.nealford.conf.canonicity.Event' alias='Event' />
  <resultMap id='eventResult' class='Event'>
    <result property='description' column='DESCRIPTION' />
    <result property='eventKey' column='EVENT_KEY' />
    <result property='start' column='START' />
    <result property='eventType' column='EVENT_TYPE' />
    <result property='duration' column='DURATION' />
  </resultMap>
  <select resultMap='eventResult' id='getEvents'>select * from event where id = ?</select>
  <select resultClass='com.nealford.conf.canonicity.Event' id='getEvent'>
    select * from event where id = #value#
  </select>
</sqlMap>
```

step 2: class builder

```
class ClassBuilder {
  def imports = []
  def fields = [:]
  def file_name
  def package_name

  def ClassBuilder(imports, fields, file_name, package_name) {
    this.imports = imports
    this.fields = fields
    this.file_name = file_name
    this.package_name = package_name
  }

  def write_imports(w) {
    imports.each { i ->
      w.writeLine("import ${i};")
    }
    w.writeLine("")
  }
}
```

```

def write_classname(w) {
    def class_name_with_extension = file_name.substring(
        file_name.lastIndexOf("/") + 1, file_name.length());
    w.writeLine("public class " +
        class_name_with_extension.substring(0,
            class_name_with_extension.length() - 5) + " {")
}

def write_fields(w) {
    fields.each { name, type ->
        w.writeLine("\t$type $name;");
    }
    w.writeLine("")
}

```

```

public class Event {
    String description;
    int eventKey;
    String start;
    int eventType;
    int duration;
}

```

```

def write_properties(w) {
    fields.each { name, type ->
        def cap_name = name.charAt(0).toString().toUpperCase() +
            name.substring(1)
        w.writeLine("\tpublic ${type} get${cap_name}() {")
        w.writeLine("\t\treturn ${name};\n\t}\n");

        w.writeLine("\tpublic void set${cap_name}(${type} ${name}) {")
        w.writeLine("\t\tthis.${name} = ${name};\n\t}\n")
    }
}

```

```

        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    ...
}

```

```
def generate_class_file() {
    new File(file_name).withWriter { w ->
        w.writeLine("package ${package_name};\n")
        write_imports(w)
        write_classname(w)
        write_fields(w)
        write_properties(w)
        w.writeLine("}")
    }
}
```

```
public class Event {
    String description;
    int eventKey;
    String start;
    int eventType;
    int duration;

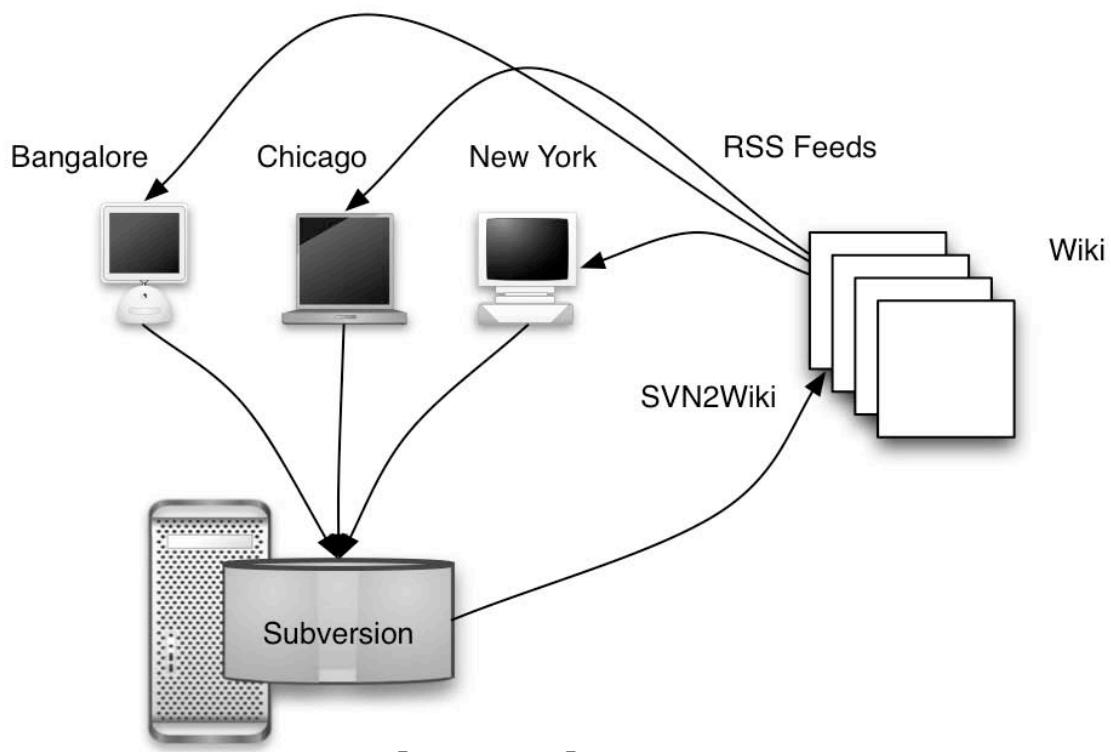
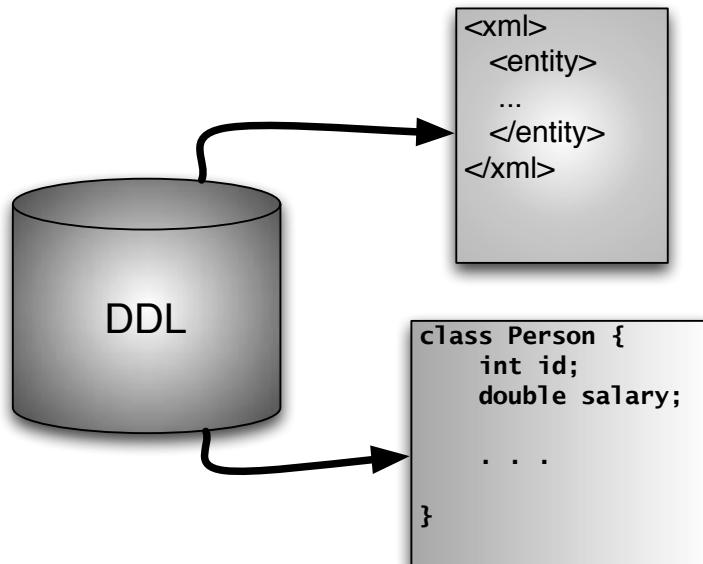
    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public int getEventKey() {
        return eventKey;
    }

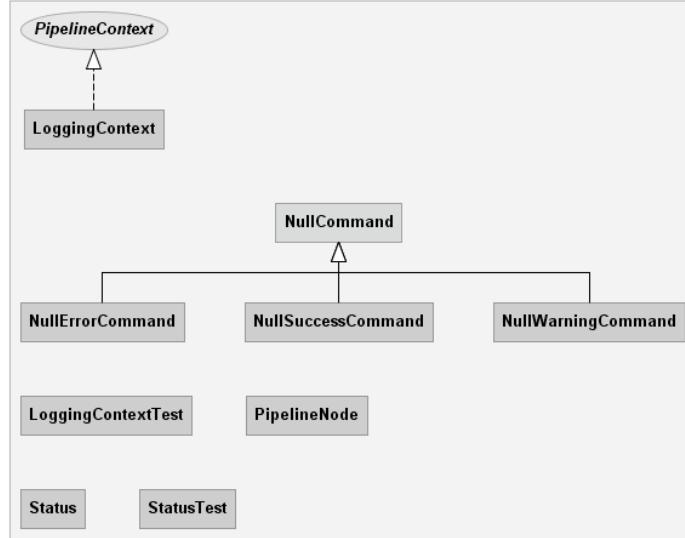
    public void setEventKey(int eventKey) {
```

canonical representation



dry documentation

dry diagrams



Generated by yDoc Evaluation Version

dry schemas

the requirement: entity-relationship diagrams
for each iteration

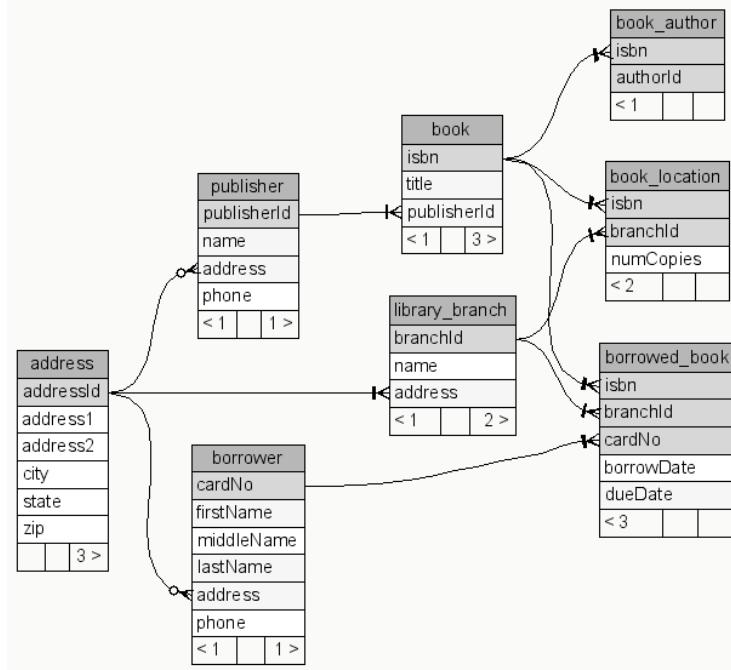
schemaspy

open source schema diagrammer

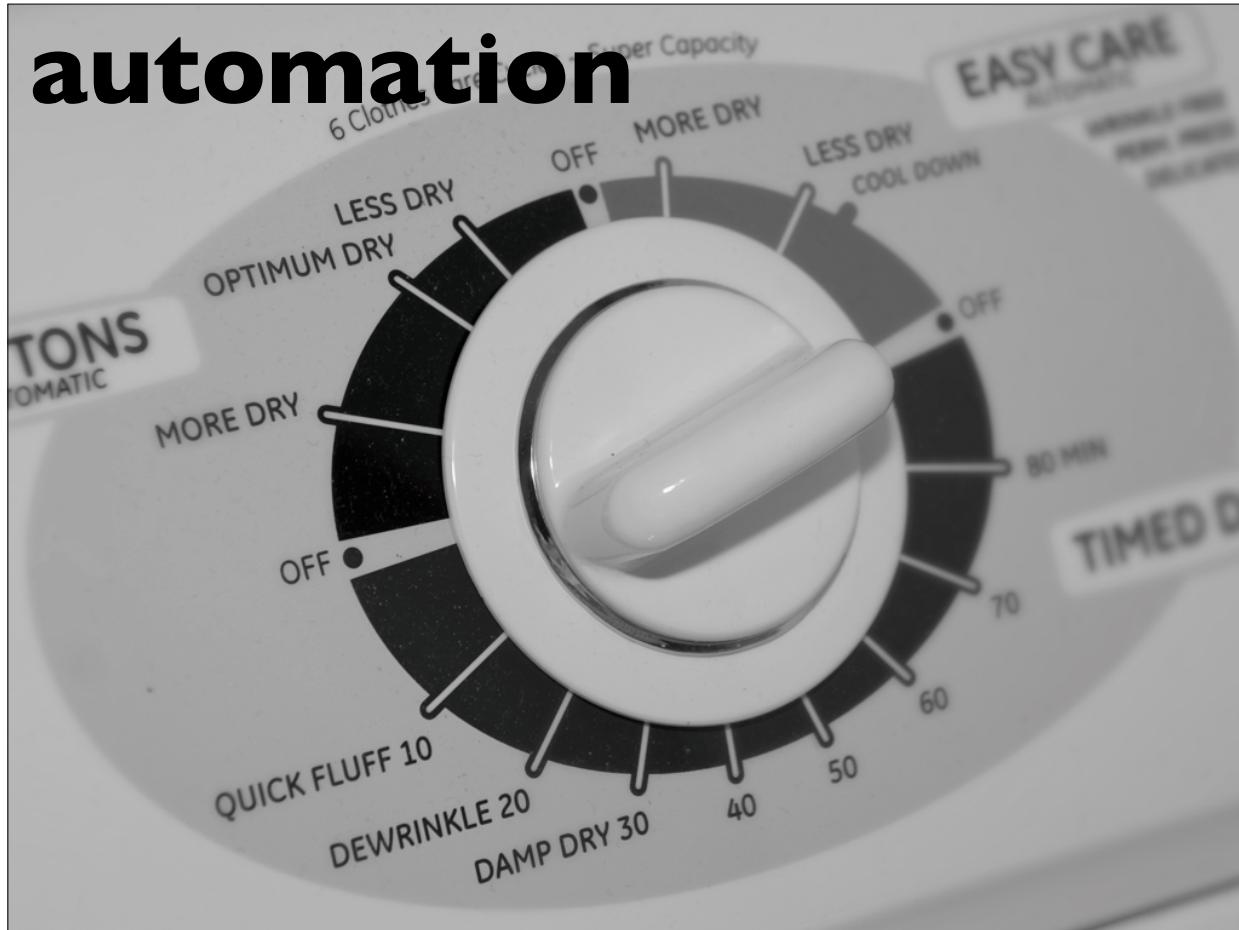
written in java

generates acceptable html

dry schemas



automation



obvious automatables

one-command build

continuous integration

version control (!)

documentation

don't build what you don't
have to build



buildix



open source project from **ThoughtWorks**

infrastructure in a box

live cd or ubuntu installation

buildix parts

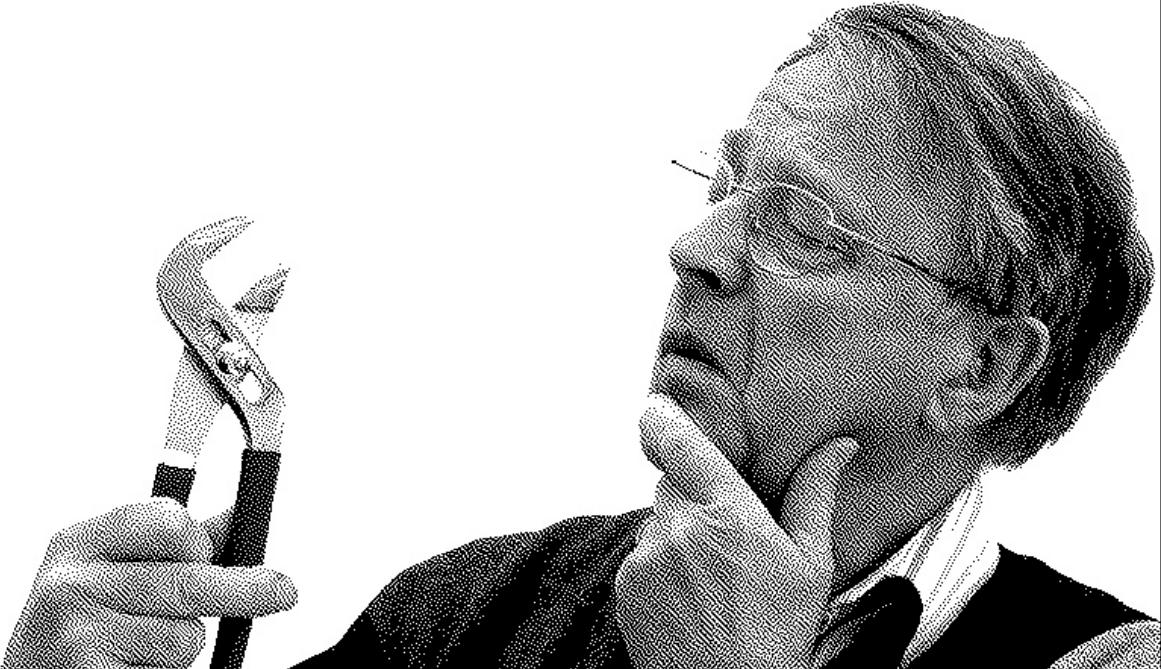
subversion

cruisecontrol

trac

mingle

subverting other tools



selenium

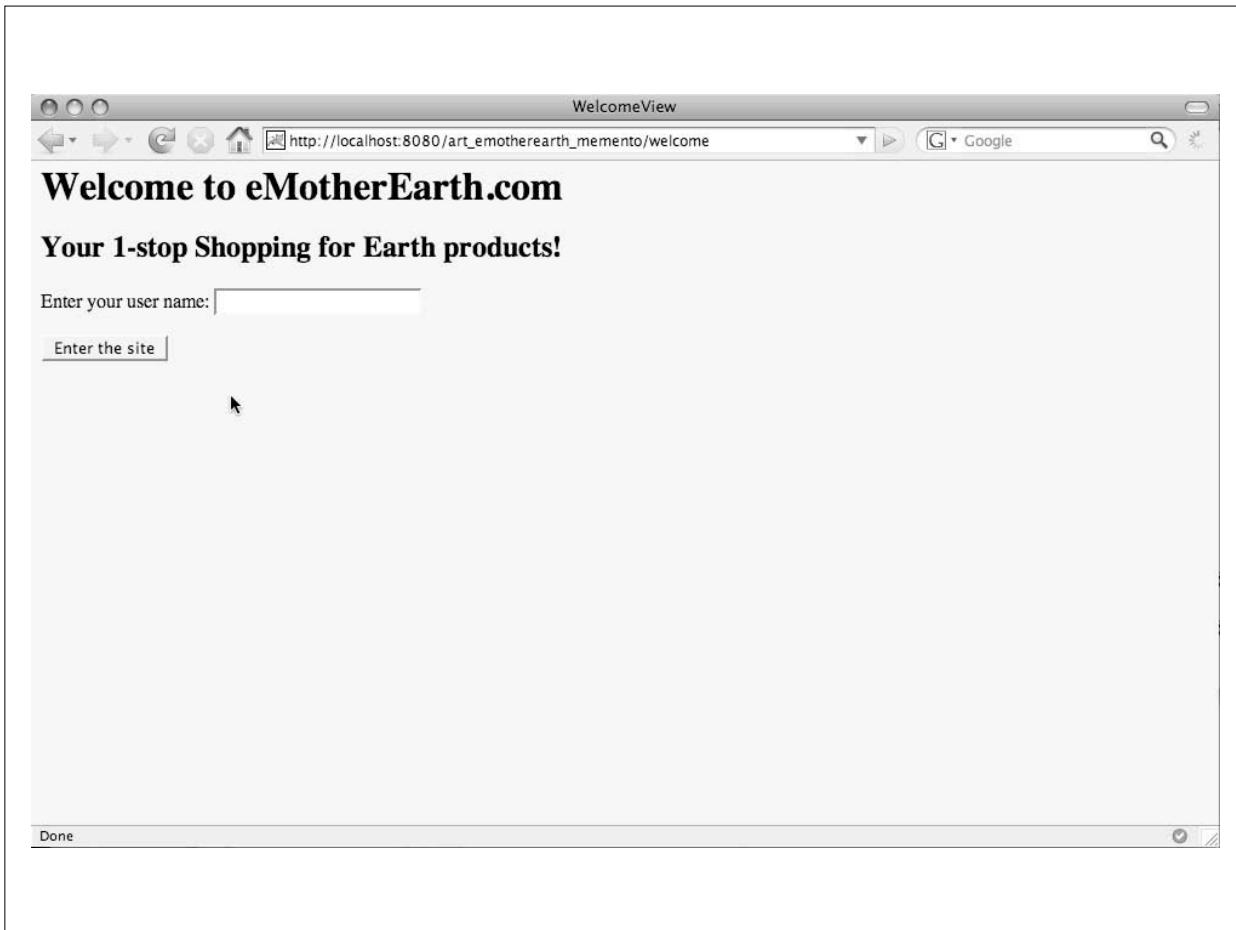
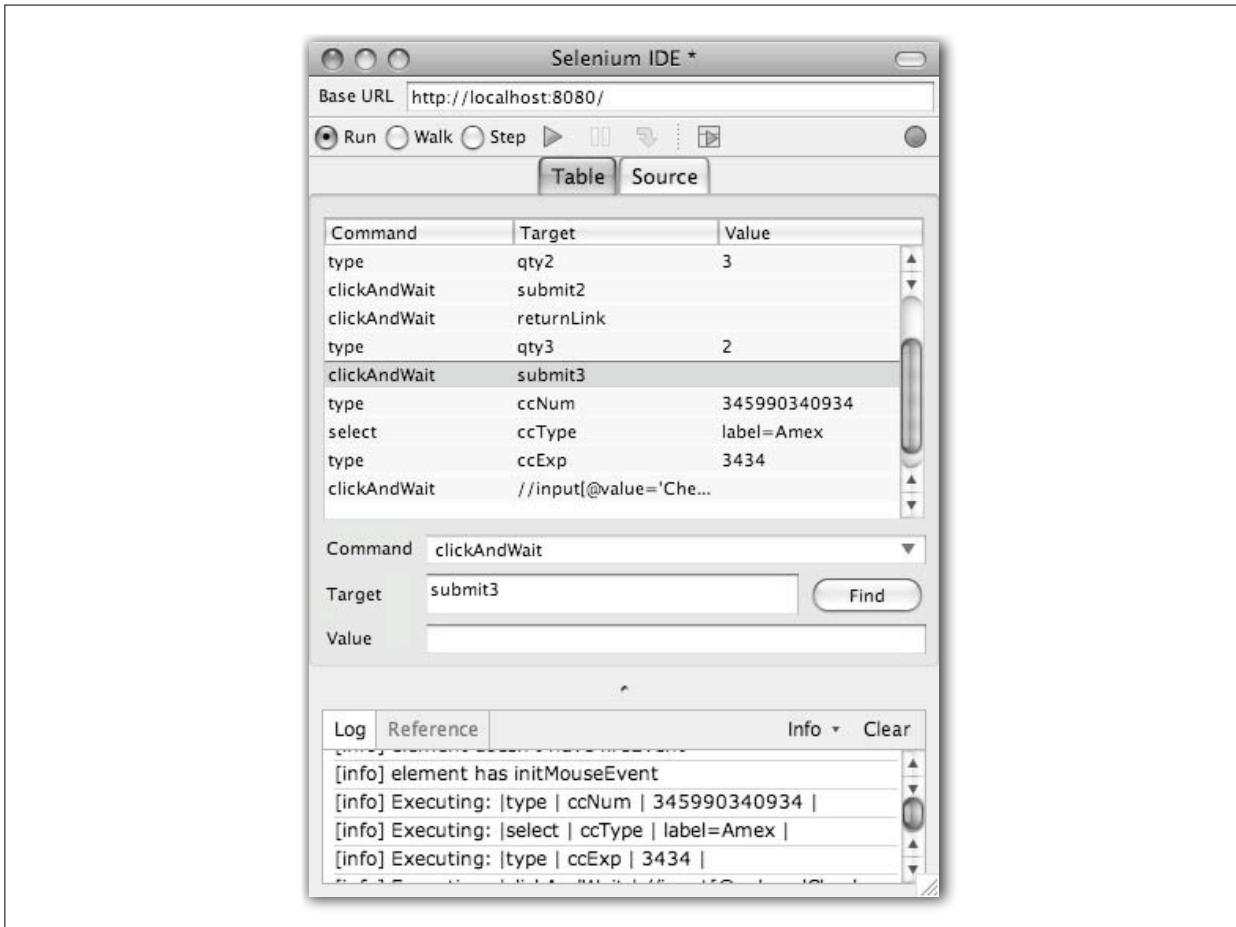
open source tool for user acceptance testing of web applications

includes a side-project called selenium ide

allows you to automate debugging “wizard”-style web applications

you always think “this is the last time”...

...but it never is!



New Test		
open	/art_emotherearth_memento/welcome	
type	userName	Homer
clickAndWait	submitButton	
type	qty2	3
clickAndWait	submit2	
clickAndWait	returnLink	
type	qty6	4
clickAndWait	submit6	
type	ccNum	234234234234
select	ccType	label=MC
type	ccExp	2323
clickAndWait	//input[@value='Check out']	

```

public class NewTest extends SeleneseTestCase {
    public void testNew() throws Exception {
        selenium.open("/art_emotherearth_memento/welcome");
        selenium.type("userName", "Homer");
        selenium.click("submitButton");
        selenium.waitForPageToLoad("30000");
        selenium.type("qty2", "3");
        selenium.click("submit2");
        selenium.waitForPageToLoad("30000");
        selenium.click("returnLink");
        selenium.waitForPageToLoad("30000");
        selenium.type("qty6", "4");
        selenium.click("submit6");
        selenium.waitForPageToLoad("30000");
        selenium.type("ccNum", "234234234234");
        selenium.select("ccType", "label=MC");
        selenium.type("ccExp", "2323");
        selenium.click("//input[@value='Check out']");
        selenium.waitForPageToLoad("30000");
    }
}

```

```

class NewTest < Test::Unit::TestCase
  def setup
    @verification_errors = []
    if $selenium
      @selenium = $selenium
    else
      @selenium = Selenium::SeleneseInterpreter.new(
        "localhost", 4444, "*firefox", "http://localhost:4444", 10000);
      @selenium.start
    end
    @selenium.set_context("test_new", "info")
  end

  def teardown
    @selenium.stop unless $selenium
    assert_equal [], @verification_errors
  end

  def test_new
    @selenium.open "/art_emotetherearth_memento/welcome"
    @selenium.type "userName", "Homer"
    @selenium.click "submitButton"
    @selenium.wait_for_page_to_load "30000"
    @selenium.type "qty2", "3"
    @selenium.click "submit2"
    @selenium.wait_for_page_to_load "30000"
    @selenium.click "returnLink"
    @selenium.wait_for_page_to_load "30000"
    @selenium.type "qty6", "4"
    @selenium.click "submit6"
    @selenium.wait_for_page_to_load "30000"
    @selenium.type "ccNum", "234234234234"
    @selenium.select "ccType", "label=MC"
    @selenium.type "ccExp", "2323"
    @selenium.click "//input[@value='Check out']"
    @selenium.wait_for_page_to_load "30000"
  end
end

```

automated interaction

record your interaction the 1st time you walk through the page

literally cuts hours off debugging time

selenium defines an interaction api for web applications

have your q/a department record bug discoveries

**don't spend time doing by
hand what you can automate**



build your own tools



you almost never do anything just once

work like a craftsman, not a laborer

build shims & jigs

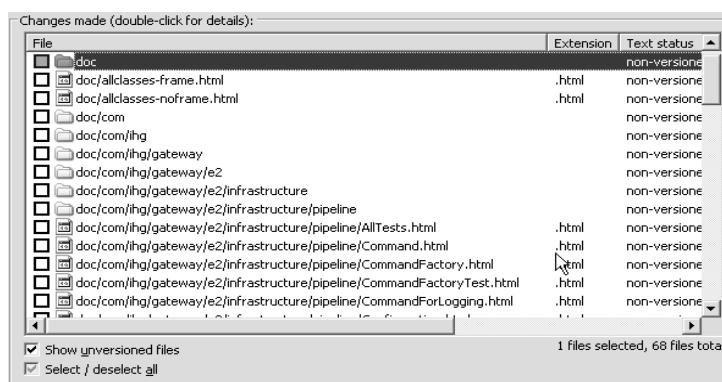
building a tool takes a little longer than brute force...

...but you build assets

bash-fu

adding new files to subversion repository

tortoise (on windows), but with limits



svnAddNew

```
svn st | grep '^?*' | tr '?*' ' ' |  
sed 's/ [ ]*//' | sed 's/[ ]/\ \ /g' | xargs svn add
```

svn st

get svn status (new files start with "?")

grep '^?*

find all new files

tr '?*' ' '

translate the "?" into a space

sed 's/[]*//'

substitute spaces to nothing

sed 's/[]/\ \ /g'

escape embedded spaces

xargs svn add

pipe the improved arguments into svn

more bash-fu

the problem: 2 gb of log files / week

need to know the count of each exception type

by hand?!!

automate with a bash script

```

get all exception types from log file
↓
#!/bin/bash
for X in $(egrep -o "[A-Z]\w*Exception" genesis_week.txt | sort | uniq) ;
do
    echo -n -e "$X\t"
    grep -c "$X" genesis_week.txt
done

```

sort them

↓

get unique list

↑
get counts of each
exception

automating com

```

def open_daily_logs
    excel = WIN32OLE.new("excel.application")

    workbooks = excel.WorkBooks
    excel.Visible = true
    doc_list.each do |f|
        begin
            workbooks.Open(@@Home_Dir + f, true)
        rescue
            puts "Cannot open workbook:", @@Home_Dir + f
        end
    end
    excel.Windows.Arrange(7)
end

```

scripting rationale

examples in lots of different languages/tools

which one do I use for this problem?

use a *real* language for scripting

sql splitter

the problem: split a 38,000 line sql file into
1000 line chunks

each chunk must be syntactically correct

“we can do it by hand in 10 minutes...”

automate instead

after 50 minutes:

```
SQL_FILE = "./GeneratedTestData.sql"
OUTPUT_PATH = "./chunks of sql/"

line_num = 1
file_num = 0
Dir.mkdir(OUTPUT_PATH) unless File.exists? OUTPUT_PATH
file = File.new(OUTPUT_PATH + "chunk " + file_num.to_s + ".sql",
  File::CREAT|File::TRUNC|File::RDWR, 0644)

done, seen_1k_lines = false
IO.readlines(SQL_FILE).each do |line|
  file.puts(line)
  seen_1k_lines = (line_num % 1000 == 0) unless seen_1k_lines
  line_num += 1
  done = (line.downcase =~ /^go$/ or
    line.downcase =~ /end$/) != nil
  if done and seen_1k_lines
    file_num += 1
    file = File.new(OUTPUT_PATH + "chunk " + file_num.to_s + ".sql",
      File::CREAT|File::TRUNC|File::RDWR, 0644)
    done, seen_1k_lines = false
  end
end
```

time spent automating

it took us 5 times longer to automate it

we've had to do it numerous times since

it “accidentally” became an important part of
our project

using a real language allowed us to refactor it...

...so that we could write unit tests

```

def test_mocked_out_dir
  ss = SqlSplitter.new("dummy_path", "dummy_file")
  Dir.expects(:mkdir).with("dummy_path")
  ss.make_a_place_for_output_files
end

def test_that_output_directory_is_created_correctly
  ss = SqlSplitter.new(OUTPUT_PATH, nil)
  ss.make_a_place_for_output_files
  assert File.exists? OUTPUT_PATH
end

def test_that_lines_o_sql_has_lines_o_sql
  lines = %w{Lorem ipsum dolor sit amet consectetur}
  ss = SqlSplitter.new(nil, nil)
  ss.sql_lines = lines
  assert ss.lines_o_sql.size > 0
  assert_same ss.lines_o_sql, lines
end

def test_generate_sql_chunks
  ss = SqlSplitter.new(OUTPUT_PATH, nil)
  ss.sql_lines = lots_o_fake_data
  ss.generate_sql_chunks
  assert File.exists? OUTPUT_PATH
  assert Dir.entries(OUTPUT_PATH).size > 0
  Dir.entries(OUTPUT_PATH).each do |f|
    assert f.size > 0
  end
end

```

using real languages

allow throw-aways to grow into assets

allows unit testing, refactoring, ide support

if you start by treating it as a 1st class problem,
you'll build better solutions

time savings

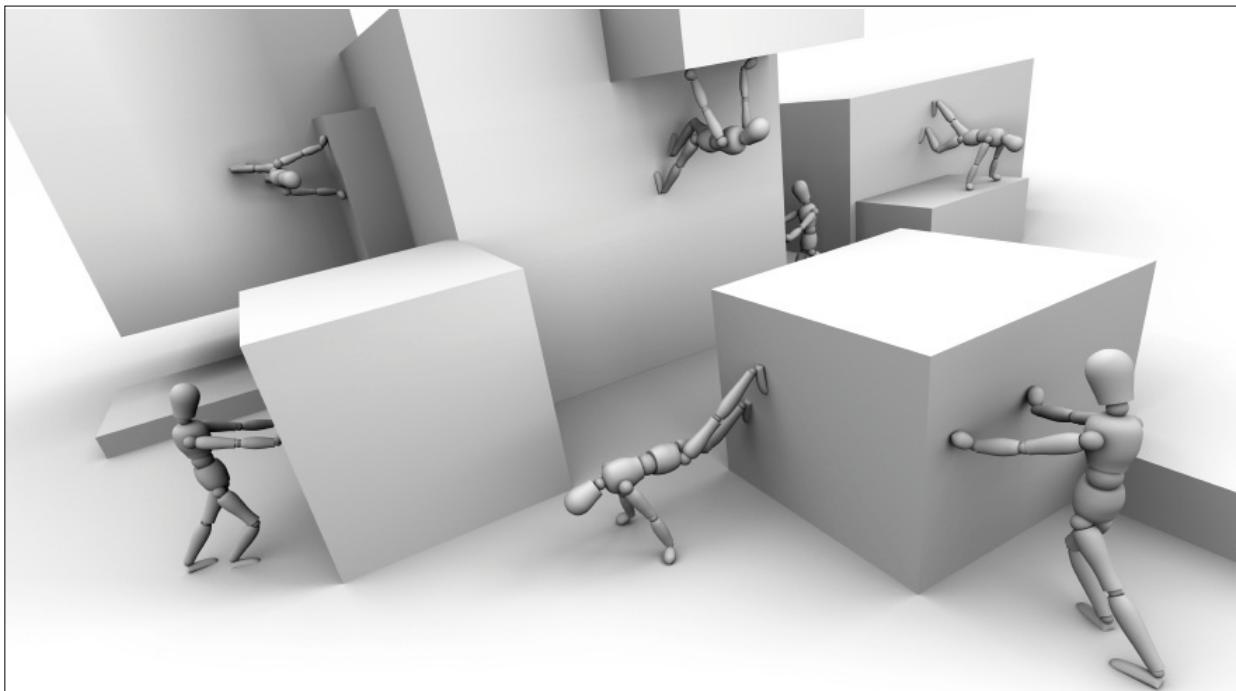
solving problems by hand makes you dumber

steals concentration

squanders focus

automating makes you smarter

figure out clever ways to solve problems



justifying automation

timebox

set a reasonable time to see if it's possible

evaluate at the end of the box

decide if you want to go forward

or create another time box

or abandon the effort

analyze the r.o.i.

how long does it take now X # of times we must do it?

what are the consequences of doing it wrong 1 time?

automation is about

time savings

risk mitigation



ThoughtWorks

questions?

please fill out the session evaluations
slides & samples available at nealford.com



This work is licensed under the Creative Commons
Attribution-Noncommercial-Share Alike 2.5 License.

<http://creativecommons.org/licenses/by-nc-sa/2.5/>

NEAL FORD software architect / meme wrangler

ThoughtWorks

nford@thoughtworks.com
3003 Summit Boulevard, Atlanta, GA 30319
www.nealford.com
www.thoughtworks.com
memeagora.blogspot.com

N

resources

The Productive Programmer

© 2008, Neal Ford

Published by O'Reilly Media

ISBN: 978-0-596-51978-0

Photos by Candy Ford

NF

