the productive programmer
where did this topic come from?
where did this topic come from?
where did this topic come from?
part 1: mechanics

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????

cicl

iClip ($$)
mac os x : jumpcut
demo
context switching eats time
there and back

`pushd` pushes a directory on the stack

`popd` pops it back off
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
try {
    s = c.createStatement();
    transactionState = c.setAutoCommit();
    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) {
    }
}
learning shortcuts

repeat them to yourself

flash cards

create a “cheat sheet”
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);
    }
}
goto class: pattern of capital letters
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) throws SQLException {
    Iterator lineItemIter = cart.getItemList().iterator();
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());
    }
}
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());
    }
}

introduce variable redux
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) throws SQLException {
    Iterator<CartItem> it = cart.getLineItemList().iterator();
# some choice shortcuts

<table>
<thead>
<tr>
<th>Feature</th>
<th>IntelliJ</th>
<th>Eclipse</th>
</tr>
</thead>
<tbody>
<tr>
<td>goto class</td>
<td>ctrl-n</td>
<td>ctrl-shift-t</td>
</tr>
<tr>
<td>introduce variable</td>
<td>ctrl-alt-v</td>
<td>alt-shift-l</td>
</tr>
<tr>
<td>escalating selection</td>
<td>ctrl-w</td>
<td>alt-shift-up</td>
</tr>
<tr>
<td>recently edited files</td>
<td>ctrl-e</td>
<td>n/a (ctrl-e)</td>
</tr>
<tr>
<td>symbol list</td>
<td>alt-ctrl-shift-n</td>
<td>ctrl-o</td>
</tr>
<tr>
<td>incremental search</td>
<td>alt-f3</td>
<td>ctrl-j</td>
</tr>
</tbody>
</table>
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
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

geta 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
internet blockers

http://getconcentrating.com/

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”
This is not the solution!

Puma Productivity Pants™
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 desktop manager power toy

http://virtuawin.sourceforge.net/

spaces (in leopard)
Click a desktop, and then select the image you would like to use as its background.

Background:
- Picture2.jpg
- Power.jpg
- Purple flower.jpg
- Radiance.jpg
- Red moon desert.jpg
- Ripple.jpg
- step1_focus.gif
- step2.gif
- Stonehenge.jpg
- Tulips.jpg
- Vortec space.jpg

Position:
- Stretch

Browse...
canonicality
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 > 1

decide on the canonical representation

generate the others
the scenario

where's the information?
canonical representation

```java
class Person {
    int id;
    double salary;
    ...
}
```
<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>
</sqlMap>

<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>
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.getColumnLabel(index)
        types.put(camelize(rsmd.getColumnLabel(index)),
                  rsmd.getColumnClassName(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.canonicality.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.canonicality.Event',
            "select * from event where id = #{value#}")
    }

    new File(_file_name).withWriter { w ->
        w.writeLine("${writer.toString()}")
    }
}
<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>
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.writeln("public class "+
        class_name_with_extension.substring(0,
            class_name_with_extension.length() - 5) + " {")
}

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

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\t\n\t};")

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

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

class Person {
    int id;
    double salary;
    ...
}

<xml>
    <entity>
        ...
    </entity>
</xml>
dry documentation
dry diagrams
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
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!
Welcome to eMotherEarth.com

Your 1-stop Shopping for Earth products!

Enter your user name: 

[Enter the site]
<table>
<thead>
<tr>
<th>Action</th>
<th>Element/Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>/art_emotherearth_memento/welcome</td>
</tr>
<tr>
<td>type</td>
<td>userName</td>
</tr>
<tr>
<td></td>
<td>homer</td>
</tr>
<tr>
<td>clickAndWait</td>
<td>submitButton</td>
</tr>
<tr>
<td>type</td>
<td>qty2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td>clickAndWait</td>
<td>submit2</td>
</tr>
<tr>
<td>clickAndWait</td>
<td>returnLink</td>
</tr>
<tr>
<td>type</td>
<td>qty6</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td>clickAndWait</td>
<td>submit6</td>
</tr>
<tr>
<td>type</td>
<td>ccNum</td>
</tr>
<tr>
<td></td>
<td>234234234234234</td>
</tr>
<tr>
<td>select</td>
<td>ccType</td>
</tr>
<tr>
<td></td>
<td>label=MC</td>
</tr>
<tr>
<td>type</td>
<td>ccExp</td>
</tr>
<tr>
<td></td>
<td>2323</td>
</tr>
<tr>
<td>clickAndWait</td>
<td>//input[@value='Check out']</td>
</tr>
</tbody>
</table>
public class NewTest extends SeleneseTestCase {
    public void testNew() throws Exception {
        selenium.open("/art_emotetherearth_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_emotherearth_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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>svn st</code></td>
<td>Get SVN status (new files start with “?”)</td>
</tr>
<tr>
<td><code>grep '^\?'</code></td>
<td>Find all new files</td>
</tr>
<tr>
<td><code>tr '^\?' ' '</code></td>
<td>Translate the “?” into a space</td>
</tr>
<tr>
<td><code>sed 's/[ ]*///'</code></td>
<td>Substitute spaces to nothing</td>
</tr>
<tr>
<td><code>sed 's/[ ]//g'</code></td>
<td>Escape embedded spaces</td>
</tr>
<tr>
<td><code>xargs svn add</code></td>
<td>Pipe the improved arguments into SVN</td>
</tr>
</tbody>
</table>
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
#!/bin/bash
for X in $(egrep -o "[A-Z]\w*Exception" genesis_week.txt | sort | uniq);
  do
echo -n -e "\n"
grep -c "$X" genesis_week.txt
  done
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.open(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 =~ /^\W*go\W*$/ or
           line.downcase =~ /^\W*end\W*$/) != nil
  if done and seen_1k_lines
    file_num += 1
    file = File.open(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
```ruby
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
don’t shave yaks!
end of part 1:
mechanics

next: practice
10 ways to improve your code
composed method
composed method

Divide your program into methods that perform one identifiable task.

Keep all of the operations in a method at the same level of abstraction.

This will naturally result in programs with many small methods, each a few lines long.
refactoring to composed method
public void populate() throws Exception {
    Connection c = null;
    try {
        Class.forName(DRIVER_CLASS);
        c = DriverManager.getConnection(DB_URL, USER, PASSWORD);
        Statement stmt = c.createStatement();
        ResultSet rs = stmt.executeQuery(SQL_SELECT_PARTS);
        while (rs.next()) {
            Part p = new Part();
            p.setName(rs.getString("name"));
            p.setBrand(rs.getString("brand"));
            p.setRetailPrice(rs.getDouble("retail_price"));
            partList.add(p);
        }
    } finally {
        c.close();
    }
}
private void addPartToListFromResultSet(ResultSet rs) throws SQLException {
    Part p = new Part();
    p.setName(rs.getString("name"));
    p.setBrand(rs.getString("brand"));
    p.setRetailPrice(rs.getDouble("retail_price"));
    partList.add(p);
}

public void populate() throws Exception {
    Connection c = null;
    try {
        c = getDatabaseConnection();
        ResultSet rs = createResultSet(c);
        while (rs.next())
            addPartToListFromResultSet(rs);
    } finally {
        c.close();
    }
}

private ResultSet createResultSet(Connection c) throws SQLException {
    return c.createStatement().executeQuery("SQL_SELECT_PARTS");
}

private Connection getDatabaseConnection() throws ClassNotFoundException, SQLException {
    Connection c;
    Class.forName("DRIVER_CLASS");
    c = DriverManager.getConnection("DB_URL", "webuser", "webpass");
    return c;
}
private Connection getDatabaseConnection() throws ClassNotFoundException, SQLException {
    Connection c;
    Class.forName(DRIVER_CLASS);
    c = DriverManager.getConnection(DB_URL, "webuser", "webpass");
    return c;
}
private ResultSet createResultSet(Connection c) throws SQLException {
    return c.createStatement().
    executeQuery("SQL_SELECT_PARTS");
}
**BoundaryBase**

```java
abstract protected String getSqlForEntity();

protected ResultSet createResultSet(Connection c) throws SQLException {
    Statement stmt = c.createStatement();
    return stmt.executeQuery(getSqlForEntity());
}
```

**PartDb**

```java
private ResultSet createResultSet(Connection c) throws SQLException {
    return c.createStatement().executeQuery(SQL_SELECT_PARTS);
}
```
BoundaryBase
getDatabaseConnection()
getSqlForEntity()
createResultSet()

PartDb
populate()
getSqlForEntity()
addPartToListFromResultSet()
public void populate() throws Exception {
    Connection c = null;
    try {
        c = getDatabaseConnection();
        ResultSet rs = createResultSet(c);
        while (rs.next())
            addPartToListFromResultSet(rs);
    } finally {
        c.close();
    }
}
abstract protected void addEntityToListFromResultSet(ResultSet rs) throws SQLException;

public void populate() throws Exception {
    Connection c = null;
    try {
        c = getDatabaseConnection();
        ResultSet rs = createResultSet(c);
        while (rs.next())
            addEntityToListFromResultSet(rs);
    } finally {
        c.close();
    }
}
protected Connection getDatabaseConnection() throws ClassNotFoundException, SQLException {
    Connection c;
    Class.forName(DRIVER_CLASS);
    c = DriverManager.getConnection(DB_URL, "webuser", "webpass");
    return c;
}

abstract protected String getSqlForEntity();

protected ResultSet createResultSet(Connection c) throws SQLException {
    Statement stmt = c.createStatement();
    return stmt.executeQuery(getSqlForEntity());
}

abstract protected void addEntityToListFromResultSet(ResultSet rs)
    throws SQLException;

public void populate() throws Exception {
    Connection c = null;
    try {
        c = getDatabaseConnection();
        ResultSet rs = createResultSet(c);
        while (rs.next())
            addEntityToListFromResultSet(rs);
    } finally {
        c.close();
    }
}
public Part[] getParts() {
    return (Part[]) partList.toArray(TEMPLATE);
}

protected String getSqlForEntity() {
    return SQL_SELECT_PARTS;
}

protected void addEntityToListFromResultSet(ResultSet rs) throws SQLException {
    Part p = new Part();
    p.setName(rs.getString("name"));
    p.setBrand(rs.getString("brand"));
    p.setRetailPrice(rs.getDouble("retail_price"));
    partList.add(p);
}
benefits of composed method

shorter methods easier to test

method names become documentation

large number of very cohesive methods

discover reusable assets that you didn’t know were there
test-driven development

2

test-driven design
design benefits of tdd

first consumer

think about how the rest of the world uses this class

creates consumption awareness
design benefits of tdd

forces mocking of dependent objects

naturally creates composed method

cleaner metrics
extroverted object
Customer

addOrder(Order o)

Order

introverted object
extroverted objects

“reach out” to create objects
ad hoc creation

introverted objects

cleaner dependencies
moves object construction to a few simple places
3 static analysis
byte-code analysis: findbugs
bug categories

correctness
  probable bug

bad practice
  violation of recommended & essential coding practice

dodgy
  confusing, anomalous, written poorly
Equals method for ApplicationMap$1 assumes the argument is of type ApplicationMap$1
At ApplicationMap.java:[line 88]
In method org.apache.struts2.dispatcher.ApplicationMap$1.equals(Object) [Lines 88 - 90]

Equals method should not assume anything about the type of its argument
The equals(Object o) method shouldn't make any assumptions about the type of o. It should simply return false if o is not the same type as this.
Impossible cast

This cast will always throw a ClassCastException.
source analysis & pmd
pmd targets

possible bugs
- empty try/catch blocks
- unused local variables
- parameters
- private variables
- wasteful string usage

dead code

suboptimal code

overcomplicated expressions
<table>
<thead>
<tr>
<th>File Path</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>hibernate/query/QueryTranslator.java</td>
<td>401</td>
</tr>
<tr>
<td>hibernate/query/WhereParser.java</td>
<td>134</td>
</tr>
<tr>
<td>hibernate/query/WhereParser.java</td>
<td>136</td>
</tr>
<tr>
<td>hibernate/test/Child.java</td>
<td>38</td>
</tr>
<tr>
<td>hibernate/test/Child.java</td>
<td>38</td>
</tr>
<tr>
<td>hibernate/test/FooComponent.java</td>
<td>76</td>
</tr>
<tr>
<td>hibernate/test/FooComponent.java</td>
<td>79</td>
</tr>
<tr>
<td>hibernate/test/Fum.java</td>
<td>46</td>
</tr>
<tr>
<td>hibernate/test/Qux.java</td>
<td>78</td>
</tr>
<tr>
<td>hibernate/test/Qux.java</td>
<td>86</td>
</tr>
<tr>
<td>hibernate/test/Qux.java</td>
<td>93</td>
</tr>
<tr>
<td>hibernate/test/Qux.java</td>
<td>100</td>
</tr>
<tr>
<td>hibernate/test/Qux.java</td>
<td>108</td>
</tr>
<tr>
<td>hibernate/test/Qux.java</td>
<td>149</td>
</tr>
<tr>
<td>hibernate/test/Qux.java</td>
<td>153</td>
</tr>
<tr>
<td>hibernate/tools/SchemaExport.java</td>
<td>191</td>
</tr>
<tr>
<td>hibernate/tools/SchemaExportTask.java</td>
<td>56</td>
</tr>
<tr>
<td>hibernate/tools/SchemaExportTask.java</td>
<td>59</td>
</tr>
<tr>
<td>hibernate/tools/codegen/ClassMapping.java</td>
<td>399</td>
</tr>
<tr>
<td>hibernate/tools/reflect/ReflectedClass.java</td>
<td>263</td>
</tr>
<tr>
<td>hibernate/tools/reverse/MapGui.java</td>
<td>548</td>
</tr>
<tr>
<td>hibernate/tools/reverse/MapGui.java</td>
<td>560</td>
</tr>
<tr>
<td>hibernate/tools/reverse/MapGui.java</td>
<td>570</td>
</tr>
<tr>
<td>hibernate/tools/reverse/MapGui.java</td>
<td>606</td>
</tr>
<tr>
<td>hibernate/tools/reverse/MapGui.java</td>
<td>612</td>
</tr>
<tr>
<td>hibernate/tools/reverse/MapGui.java</td>
<td>617</td>
</tr>
<tr>
<td>hibernate/tools/reverse/MapGui.java</td>
<td>641</td>
</tr>
<tr>
<td>hibernate/tools/reverse/MapGui.java</td>
<td>651</td>
</tr>
<tr>
<td>hibernate/tools/reverse/MapGui.java</td>
<td>709</td>
</tr>
<tr>
<td>hibernate/tools/reverse/MapGui.java</td>
<td>737</td>
</tr>
<tr>
<td>hibernate/tools/reverse/ParamsPanel.java</td>
<td>142</td>
</tr>
<tr>
<td>hibernate/tools/reverse/ParamsPanel.java</td>
<td>149</td>
</tr>
<tr>
<td>hibernate/tools/reverse/ParamsPanel.java</td>
<td>158</td>
</tr>
<tr>
<td>hibernate/tools/reverse/ParamsPanel.java</td>
<td>167</td>
</tr>
<tr>
<td>hibernate/type/ArrayType.java</td>
<td>19</td>
</tr>
<tr>
<td>hibernate/type/ComponentType.java</td>
<td>31</td>
</tr>
</tbody>
</table>

- avoid unused private fields such as 'NO_INTS'
- avoid unused private fields such as 'quoted'
- avoid unused private fields such as 'bracketsSinceFunction'
- avoid unused formal parameters such as 'id'
- avoid unused private methods such as 'setId(long)'
- avoid unused private methods such as 'getNull()'
- avoid unused private methods such as 'setNull(String)'
- avoid unused private methods such as 'setId(FumCompositeID)'
- avoid unused private methods such as 'setCreated(boolean)'
- avoid unused private methods such as 'setDeleted(boolean)'
- avoid unused private methods such as 'setLoaded(boolean)'
- avoid unused private methods such as 'setStored(boolean)'
- avoid unused private methods such as 'setKey(long)'
- avoid unused private methods such as 'getChildKey()'
- avoid unused private methods such as 'setChildKey(Long)'
- avoid unused private methods such as 'format(String)'
- avoid unused private fields such as 'formatSQL'
- avoid unused private fields such as 'delimiter'
- avoid unused private methods such as 'addImport(String)'
- avoid unused private methods such as 'capitalize(String)'
- avoid unused formal parameters such as 'evt'
- avoid unused formal parameters such as 'evt'
- avoid unused formal parameters such as 'evt'
- avoid unused formal parameters such as 'evt'
- avoid unused formal parameters such as 'evt'
- avoid unused formal parameters such as 'evt'
- avoid unused formal parameters such as 'evt'
- avoid unused formal parameters such as 'evt'
- avoid unused formal parameters such as 'evt'
- avoid unused formal parameters such as 'evt'
- avoid unused formal parameters such as 'evt'
- avoid unused forman parameters such as 'evt'
- avoid unused private fields such as 'buttonGroup1'
- avoid unused formal parameters such as 'evt'
- avoid unused formal parameters such as 'evt'
- avoid unused formal parameters such as 'evt'
- avoid unused formal parameters such as 'evt'
- avoid unused private fields such as 'elementClass'
- avoid unused private fields such as 'parentProperty'
Found a 102 token duplication in the following files:
C:\j2sdk1.4.1_01\src\java\util\HashMap.java
C:\j2sdk1.4.1_01\src\java\util\WeakHashMap.java

```java
    return false;
    Map.Entry e = (Map.Entry)o;
    Object k1 = e.getKey();
    Object k2 = e.getKey();
    if (k1 == k2 || (k1 != null && k1.equals(k2))) {
        Object v1 = e.getValue();
        Object v2 = e.getValue();
        if (v1 == v2 || (v1 != null && v1.equals(v2)))
            return true;
    }
    return false;
```
good citizenship
getters & setters != encapsulation
accessors/mutators

knee-jerk creating getters/setters voids encapsulation

create atomic mutators for dependent fields

“should I unit test my getters & setters?”
the new strategy

only create getters & setters when you need them for other methods

testing:

  shouldn’t have to tdd them

  they will get code coverage automatically

as easy upon use as upon creation
constructors

specific contract for how to create valid objects

how often is a blank object valid?

never!

don’t provide default constructors for domain objects

push back on frameworks that require this
static methods

Math.sqrt(25)

Math.sqrt()
mixing static + state

singleton

singleton is bad because:

mixes responsibilities

untestable

the object version of global variables
avoiding singletons

1. create a pojo for the business behavior
   simple
testable!

2. create a factory to create the pojo
   also testable
public class ConfigSingleton {
    private static ConfigSingleton myInstance;
    private Point _initialPosition;

    public Point getInitialPosition() {
        return _initialPosition;
    }

    private ConfigSingleton() {
        Dimension screenSize =
            Toolkit.getDefaultToolkit().getScreenSize();
        _initialPosition = new Point();
        _initialPosition.x = (int) screenSize.getWidth() / 2;
        _initialPosition.y = (int) screenSize.getHeight() / 2;
    }

    public static ConfigSingleton getInstance() {
        if (myInstance == null)
            myInstance = new ConfigSingleton();
        return myInstance;
    }
}
public class Configuration {
    private Point _initialPosition;

    private Configuration(Dimension screenSize) {
        _initialPosition = new Point();
        _initialPosition.x = (int) screenSize.getWidth() / 2;
        _initialPosition.y = (int) screenSize.getHeight() / 2;
    }

    public int getInitialX() {
        return _initialPosition.x;
    }

    public int getInitialY() {
        return _initialPosition.y;
    }
}
public class ConfigurationFactory {
    private static Configuration myConfig;

    public static Configuration getConfiguration() {
        if (myConfig == null) {
            try {
                Constructor cxtor[] =
                        Configuration.class.getDeclaredConstructors();
                cxtor[0].setAccessible(true);
                myConfig = (Configuration) cxtor[0].newInstance(
                        Toolkit.getDefaultToolkit().getScreenSize());
            } catch (Throwable e) {
                throw new RuntimeException("can't construct Configuration");
            }
        }
        return myConfig;
    }
}
public class TestConfigurationFactory extends TestCase {

    public void test_Creation_create_a_single_instance() {
        Configuration config1 = ConfigurationFactory.getConfiguration();
        assertNotNull(config1);
        Configuration config2 = ConfigurationFactory.getConfiguration();
        assertNotNull(config2);
        assertEquals(config1, config2);
    }

}
the worst citizen in the java world...

java.util.Calendar
5

yagni

you ain’t gonna need it
discourages gold plating

build the simplest thing that we need right now

don’t indulge in speculative development

increases software entropy

only saves time if you can guarantee you won’t have to change it later

leads to frameworks
a public plea to the java community:

please stop building frameworks!
It looks like you're trying to write a framework. Would you like to...
- discard code?
- find an open source framework instead?
- find a new job?
This is just what they need!
a cautionary tale

building a simple framework
changeability

- anticipatory design
- refactorable

rate of change

lower → higher
10 Top Corporate Code Smells
6. We have an Architect who reviews all code pre-checkin and decides whether or not to allow it into version control.

7. We can’t use any open source code because our lawyers say we can’t.

8. We use WebSphere because...(I always stop listening at this point)

9. We bought the entire tool suite (even though we only needed about 10% of it) because it was cheaper than buying the individual tools.

10. We invented our own web/persistence/messaging/caching framework because none of the existing ones was good enough.
1. There is a reason that WSAD isn’t called WHAPPY.

2. The initial estimate must be within 15% of the final cost, the post-analysis estimate must be within 10%, and the post-design estimate must be within 5%.

3. We don’t have time to write unit tests (we’re spending too much time debugging).

4. We keep all of our business logic in stored procedures...for performance reasons.

5. The only JavaDoc is the Eclipse message explaining how to change your default JavaDoc template.
question authority
angry monkeys &
christmas roasts
test names

testUpdateCacheAndVerifyThatItemExists() {
}

test_Update_cache_and_verify_that_item_exists() {
}
api's

```java
Car car = new CarImpl();
MarketingDescription desc = new MarketingDescriptionImpl();
desc.setType("Box");
desc.setSubType("Insulated");
desc.setAttribute("length", "50.5");
desc.setAttribute("ladder", "yes");
desc.setAttribute("lining type", "cork");
car.setDescription(desc);
```
fluent interfaces

```java
Car car = Car.describedAs()
    .box()
    .length(12)
    .includes(Equipment.LADDER)
    .has(Lining.CORK);
```
what stands in the way?
the javabeans specification!
what’s bad about beans?

forces you to create default constructors

creates bad citizens

harms constructor as specification

setXXX() methods return void

can’t use beans for fluent interfaces
non-intuitive
pair programming studies

after adjusting, pairs produced code 15% more slowly than individuals...
pair programming studies

...with 15% fewer defects
single level of abstraction principle
slap

keep all lines of code in a method at the same level of abstraction
jumping abstraction layers makes code hard to understand

composed method => slap

refactor to slap

even if it means single-line methods
public void addOrder(final ShoppingCart cart, String userName,
                   Order order) throws SQLException {
    Connection c = null; PreparedStatement ps = null;
    Statement s = null; ResultSet rs = null;
    boolean transactionState = false;
    try {
        c = dbPool.getConnection();
        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.setOrderKey(orderKey);
    } catch (SQLException sqlx) {
        s = c.createStatement();
        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) {
        }
    }
}
public void addOrder(final ShoppingCart cart, String userName, Order order) throws SQLException {
    Connection connection = null; PreparedStatement ps = null;
    Statement statement = null; ResultSet rs = null;
    boolean transactionState = false;
    try {
        connection = dbPool.getConnection();
        statement = connection.createStatement();
        transactionState = setupTransactionStateFor(connection, transactionState);
        addSingleOrder(order, connection, ps, userKeyFor(userName, connection));
        order.setOrderKey(generateOrderKey(statement, rs));
        addLineItems(cart, connection, order.getOrderKey());
        completeTransaction(connection);
    } catch (SQLException sqlx) {
        rollbackTransactionFor(connection);
        throw sqlx;
    } finally {
        cleanUpDatabaseResources(connection, transactionState, statement, ps, rs);
    }
}
polyglot programming
leveraging existing *platforms* with *languages* targeted at specific problems and applications
why do this?
looming problems/opportunities

massively parallel threading

use a functional language: jaskell, scala

schedule pressure

jruby on rails, grails
looming problems/opportunities

everyday coding

groovy, ruby

stop banging rocks together & get some work done!

face it:
looming problems/opportunities

writing more declarative code via **dsls**

build fluent interfaces
polyglot programming add complexity?

In the past, language == platform

now, language != platform
public class LineNumbers {
    public LineNumbers(String path) {
        File file = new File(path);
       LineNumberReader reader = null;
        try {
            reader = new LineNumberReader(new FileReader(file));
            while (reader.ready()) {
                out.println(reader.getLineNumber() + ":
                      + reader.readLine());
            
        }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                reader.close();
            } catch (IOException ignored) {
            }
        }
    }

    public static void main(String[] args) {
        new LineNumbers(args[0]);
    }
}
def number=0
new File (args[0]).eachLine { line ->
    number++
    println "$number: $line"
}
class SafeArray{
    private final Object[] _arr;
    private final int _begin;
    private final int _len;

    public SafeArray(Object[] arr, int len){
        _arr = arr;
        _begin = begin;
        _len = len;
    }

    public Object at(int i){
        if(i < 0 || i >= _len){
            throw new ArrayIndexOutOfBoundsException(i);
        }
        return _arr[_begin + i];
    }

    public int getLength(){
        return _len;
    }
}

newSafeArray arr begin len = {
  length = len;
  at i = if i < begin || i >= len then
    throw $ ArrayIndexOutOfBoundsException.new[i]
  else
    arr[begin + i];
}
9

every nuance
java’s back alleys

reflection

“reflection is slow”

no longer true

elegant solutions to problems
public class Configuration {
    private Point _initialPosition;

    private Configuration(Dimension screenSize) {
        _initialPosition = new Point();
        _initialPosition.x = (int) screenSize.getWidth() / 2;
        _initialPosition.y = (int) screenSize.getHeight() / 2;
    }

    public int getInitialX() {
        return _initialPosition.x;
    }

    public int getInitialY() {
        return _initialPosition.y;
    }
}
@Before public void setUp() {
    try {
        Constructor cxtor[] =
            Configuration.class.getDeclaredConstructors();
        cxtor[0].setAccessible(true);
        c = (Configuration) cxtor[0].newInstance(
            Toolkit.getDefaultToolkit().getScreenSize());
    } catch (Throwable e) {
        fail();
    }
}
@Test
class PositionSetCorrectlyUponInstantiation {
    public void testInitialPositionSetCorrectlyUponInstantiation() {
        Configuration specialConfig = null;
        Dimension screenSize = null;
        try {
            Constructor[] cxtor[] = null =
                Configuration.class.getDeclaredConstructors();
            cxtor[0].setAccessible(true);
            screenSize = new Dimension(26, 26);
            specialConfig = (Configuration) cxtor[0].newInstance(screenSize);
        } catch (Throwable e) {
            fail();
        }

        Point expected = new Point();
        expected.x = (int) screenSize.getWidth() / 2;
        expected.y = (int) screenSize.getHeight() / 2;
        assertEquals(expected.x, specialConfig.getInitialX());
        assertEquals(expected.y, specialConfig.getInitialY());
    }
}
regular expressions &
learn the nuances of java...

...then tell the other people on your project
anti-objects
collaborative diffusion

“The metaphor of objects can go too far by making us try to create objects that are too much inspired by the real world.”

“...an antiobject is a kind of object that appears to essentially do the opposite of what we generally think the object should be doing.”
questions?

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

An Initial Investigation of Test Driven Development in Industry - Laurie Williams, Boby George
http://collaboration.csc.ncsu.edu/laurie/Papers/TDDpaperv8.pdf

findbugs
http://findbugs.sourceforge.net/

pmd/cpd
http://pmd.sourceforge.net/

The legend of the leaning tower
http://physicsworld.com/cws/article/print/16806

AntiPatterns Catalog
http://c2.com/cgi/wiki?AntiPatternsCatalog
resources

Smalltalk Best Practice Patterns Kent Beck
Prentice Hall PTR (October 13, 1996)
ISBN-10: 013476904X

Polyglot Programming
http://memeagora.blogspot.com/2006/12/polyglot-programming.html

Optical Illusions
http://en.wikipedia.org/wiki/Optical_illusion

Collaborative Diffusion: Programming Anti-objects - A Repenning
http://www.cs.colorado.edu/~ralex/papers/PDF/OOPSLA06antiobjects.pdf
resources
pair programming

http://c2.com/cgi/wiki?PairProgramming

http://www.xprogramming.com/Practices/PracPairs.html

http://collaboration.csc.ncsu.edu/laurie/Papers/XPSardinia.PDF

http://www.cs.utah.edu/~lwilliam/Papers/ieeeSoftware.PDF
resources

The Productive Programmer

© 2008, Neal Ford

Published by O’Reilly Media

ISBN: 978-0-596-51978-0

Photos by Candy Ford