How to stop writing next year's unsustainable piece of code

Guilherme Silveira
@guilhermecaelum

aka

"ranting about crappy code"

Guilherme Silveira

guilherme.silveira@caelum.com.br

Caelum
Ensino e Inovação

2011년 11월 16일 수요일

guilherme.silveira@caelum.com.br

Caelum
Ensino e Inovação

guilherme.silveira@caelum.com.br
@guilhermecaelum

Caelum
Ensino e Inovação

guilherme.silveira@caelum.com.br
@guilhermecaelum

Caelum
Ensino e Inovação

http://www.caelum.com.br

guilherme.silveira@caelum.com.br
@guilhermecaelum

Caelum
Ensino e Inovação

http://www.caelum.com.br
http://online.caelumobjects.com

guilherme.silveira@caelum.com.br
@guilhermecaelum

Caelum
Ensino e Inovação

http://www.caelum.com.br
http://online.caelumobjects.com

- technical leader

guilherme.silveira@caelum.com.br
@guilhermecaelum

Caelum
Ensino e Inovação

http://www.caelum.com.br
http://online.caelumobjects.com

- technical leader
- high quality training

guilherme.silveira@caelum.com.br
@guilhermecaelum

Caelum
Ensino e Inovação

http://www.caelum.com.br
http://online.caelumobjects.com

- technical leader
- high quality training
- leaders in brazil

how long have we beeing developing? years?

the most important
part in a project
is writing beautiful code

the most important
part in a project
is writing good code

Implementation

Design

Architecture

architecture >=

architecture >=

design >=

architecture >=

design >=

implementation

Browser

Controller

Hibernate

Browser

Controller

JPA

Browser

Controller

JPA

file upload

Browser

Controller

JPA

memory

Browser

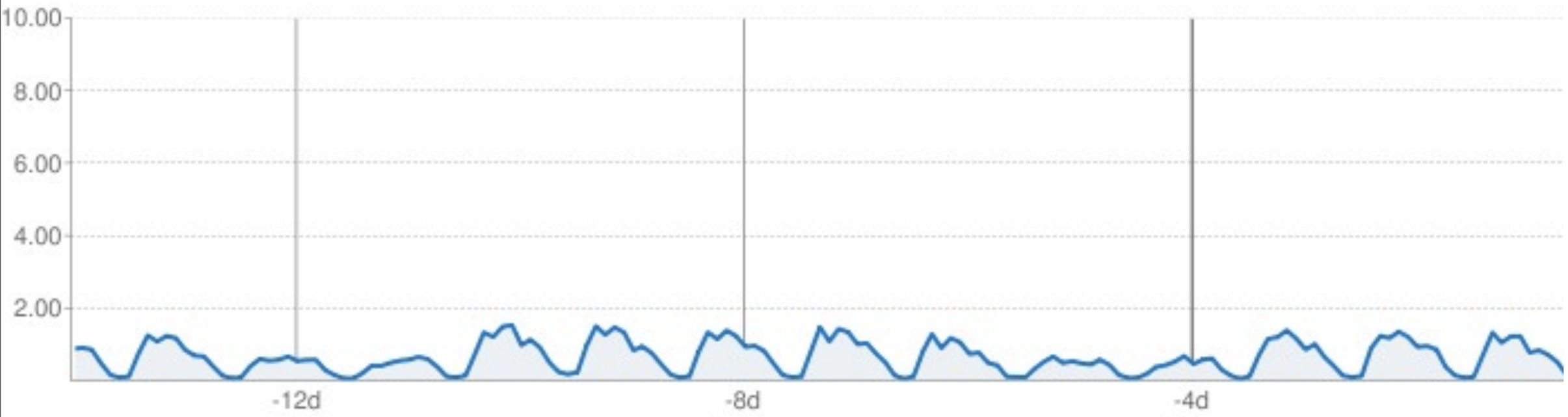
Controller

cloud

JPA

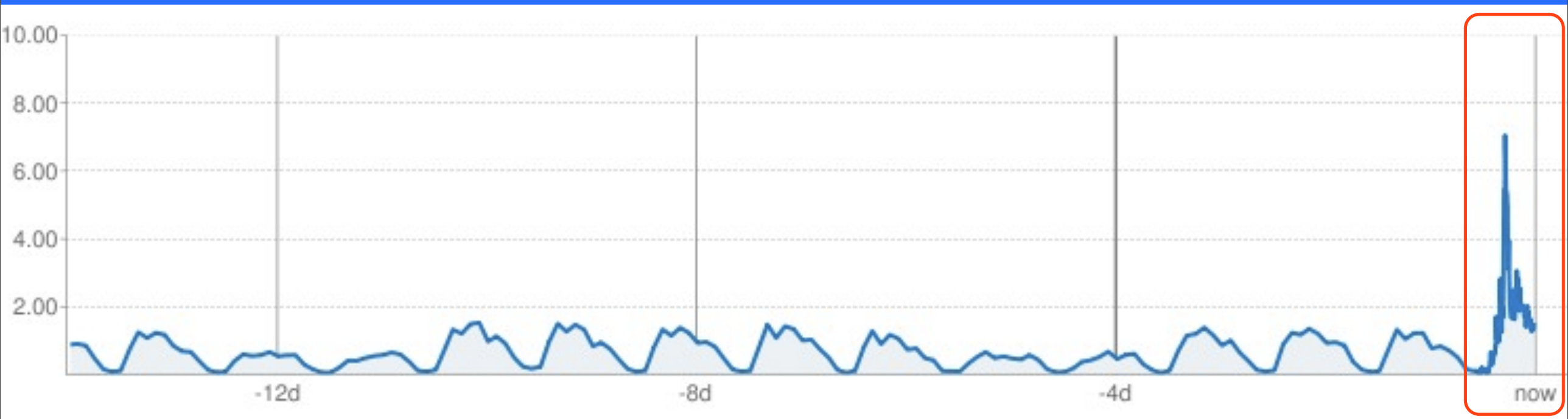
memory

14 days and



2011년 11월 16일 수요일

14 days and



BUM!

2011년 11월 16일 수요일


Cloud Computing

Implementation

Design

Architecture

IOC

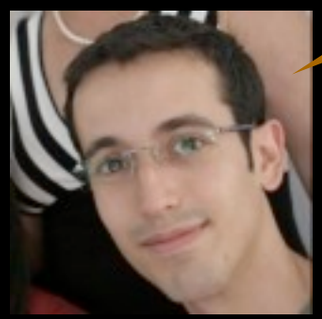
2011년 11월 16일 수요일

IOC

DESIGN

IOC

using good design practices (ioc)

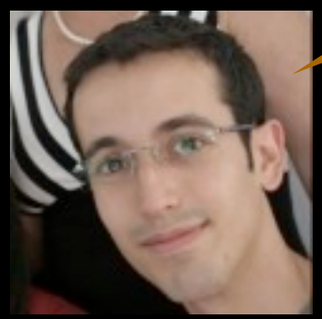


DESIGN

Paulo Silveira

IOC

using good design practices (ioc)



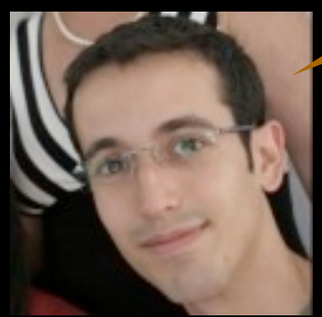
DESIGN

Paulo Silveira

ARCHITECTURE

IOC

using good design practices (ioc)

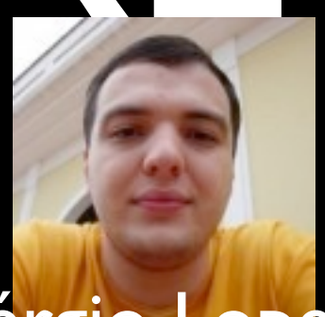


DESIGN

Paulo Silveira

switched the architecture (cloud) and increased sales

ARCHITECTURE



Sérgio Lopes

Architecture

Design

Implementation

architecture <=

architecture <=

design <=

architecture <=

design <=

implementation

architecture <=

design <=

implementation

architecture >=

architecture <=

design <=

implementation

architecture >=

design >=

architecture <=

design <=

implementation

architecture >=

design >=

implementation

architecture <=

design <=

implementation

architecture >=

design >=

implementation

architecture =

architecture <=

design <=

implementation

architecture >=

design >=

implementation

architecture =

design =

architecture <=	design <=	implementation
architecture >=	design >=	implementation
architecture =	design =	implementation

architecture

design

architecture

design

architecture

implementation

design

architecture

implementation

java, ruby, scala, objective-c, c#

servers, firewalls etc

design guys: change your design, again!

design

architecture

implementation

java, ruby, scala, objective-c, c#

servers, firewalls etc

design

architecture

implementation

java, ruby, scala, objective-c, c#

servers, firewalls etc

design guys: change your design, again!

architects: change your architecture, again!

design

architecture

implementation

java, ruby, scala, objective-c, c#

servers, firewalls etc

design guys: change your design, again!

design

architecture

implementation

java, ruby, scala, objective-c, c#

servers, firewalls etc

design guys: change your design, again!

architects: change your architecture, again!

twitter guys: change your language, again!

design

architecture

implementation

java, ruby, scala, objective-c, c#

servers, firewalls etc

design guys: change your design, again!

architects: change your architecture, again!

design

architecture

implementation

java, ruby, scala, objective-c, c#

servers, firewalls etc

twitter guys: change your language, again!
design guys: change your design, again!
architects: change your architecture, again!

there is

there is

no architecture

there is

no architecture

no design

there is

no architecture

no design

without implementation

there is implementation

the only "thing" that exists
is the implementation.

most important guys = the devs

the most important
part in a project
is writing good code

beautiful?

architecture is the minimum

if the arch is not yet good enough, big problem

2011년 11월 16일 수요일

what is left? design...

what is left? design...

interface ===> hard to maintain

what is left? design...

interface ===> hard to maintain

implementation ===> hard to maintain

DESIGN

IMPLEMENTATION

DESIGN

How does your code communicates?
(communication interface design)

IMPLEMENTATION

DESIGN

How does your code communicates?
(communication interface design)

What is the execution flow?
(implementation design)

IMPLEMENTATION

we will see code

```ruby
def _read_attribute(attr_name)
  attr_name = attr_name.to_s
  attr_name = self.class.primary_key if attr_name == 'id'
  value = @attributes[attr_name]
  unless value.nil?
    if column = column_for_attribute(attr_name)
      if unserializable_attribute?(attr_name, column)
        unserialize_attribute(attr_name)
      else
        column.type_cast(value)
      end
    else
      value
    end
  end
end
```

```ruby
def _read_attribute(attr_name)
  attr_name = attr_name.to_s
  attr_name = self.class.primary_key if attr_name == 'id'
  value = @attributes[attr_name]
  unless value.nil?
    if column = column_for_attribute(attr_name)
      if unserializable_attribute?(attr_name, column)
        unserialize_attribute(attr_name)
      else
        column.type_cast(value)
      end
    else
      value
    end
  end
end
```

too small?

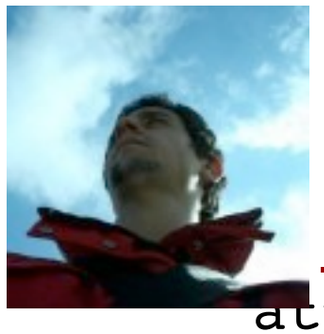
```ruby
def _read_attribute(attr_name)
  attr_name = attr_name.to_s
  attr_name = self.class.primary_key if attr_name == 'id'
  value = @attributes[attr_name]
  unless value.nil?
    if column = column_for_attribute(attr_name)
      if unserializable_attribute?(attr_name, column)
        unserialize_attribute(attr_name)
      else
        column.type_cast(value)
      end
    else
      value
    end
  end
end
```

better? font 16

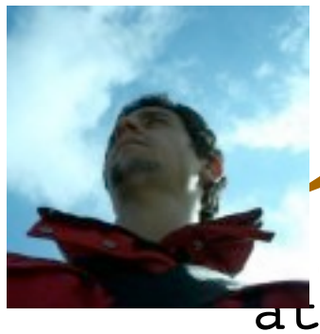
```ruby
def _read_attribute(attr_name)
  attr_name = attr_name.to_s
  attr_name = self.class.primary_key if attr_name == 'id'
  value = @attributes[attr_name]
  unless value.nil?
    if column = column_for_attribute(attr_name)
      if unserializable_attribute?(attr_name, column)
        unserialize_attribute(attr_name)
      else
        column.type_cast(value)
      end
    else
      value
    end
  end
end
```

better? font 30

```ruby
def _read_attribute(attr_name)
  attr_name = attr_name.to_s
  attr_name = self.class.primary_key if attr_name == 'id'
  value = @attributes[attr_name]
  unless value.nil?
    if column = column_for_attribute(attr_name)
      if unserializable_attribute?(attr_name, column)
        unserialize_attribute(attr_name)
      else
        column.type_cast(value)
      end
    else
      value
    end
  end
end
```

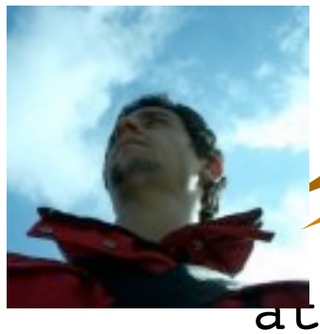


```
  _read_attribute(attr_name)
attr_name = attr_name.to_s
attr_name = self.class.primary_key if attr_name == 'id'
value = @attributes[attr_name]
unless value.nil?
  if column = column_for_attribute(attr_name)
    if unserializable_attribute?(attr_name, column)
      unserialize_attribute(attr_name)
    else
      column.type_cast(value)
    end
  else
    value
  end
 end
end
```



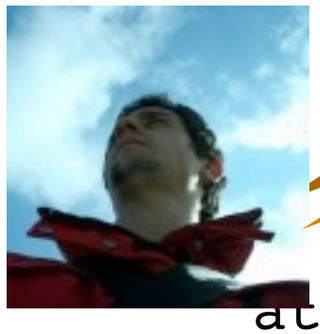
in case...

```
_read_attribute(attr_name)
attr_name = attr_name.to_s
attr_name = self.class.primary_key if attr_name == 'id'
value = @attributes[attr_name]
unless value.nil?
  if column = column_for_attribute(attr_name)
    if unserializable_attribute?(attr_name, column)
      unserialize_attribute(attr_name)
    else
      column.type_cast(value)
    end
  else
    value
  end
end
end
```



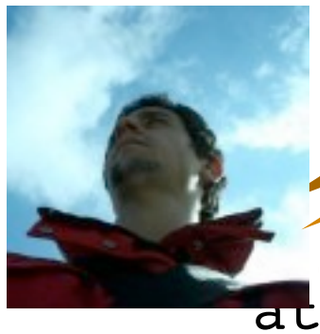
in case...

```
_read_attribute(attr_name)
attr_name = attr_name.to_s
attr_name = self.class.primary_key if attr_name == 'id'
value = @attributes[attr_name]
unless value.nil?
  if column = column_for_attribute(attr_name)
    if unserializable_attribute?(attr_name, column)
      unserialize_attribute(attr_name)
    else
      column.type_cast(value)
    end
  else
    value
  end
 end
end
```



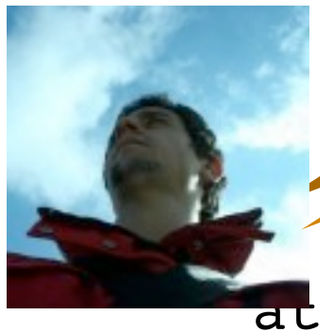
in case...

```
_read_attribute(attr_name)
attr_name = attr_name.to_s
attr_name = self.class.primary_key if attr_name == 'id'
value = @attributes[attr_name]
unless value.nil?
  if column = column_for_attribute(attr_name)
    if unserializable_attribute?(attr_name, column)
      unserialize_attribute(attr_name)
    else
      column.type_cast(value)
    end
  else
    value
  end
end
end
```



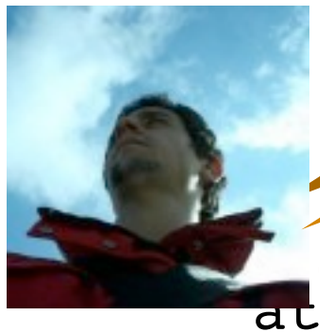
in case...

```
_read_attribute(attr_name)
attr_name = attr_name.to_s
attr_name = self.class.primary_key if attr_name == 'id'
value = @attributes[attr_name]
unless value.nil?
  if column = column_for_attribute(attr_name)
    if unserializable_attribute?(attr_name, column)
      unserialize_attribute(attr_name)
    else
      column.type_cast(value)
    end
  else
    value
  end
 end
end
```



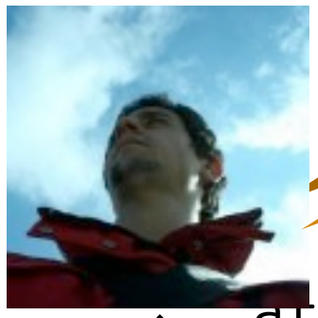
in case...

```
_read_attribute(attr_name)
attr_name = attr_name.to_s
attr_name = self.class.primary_key if attr_name == 'id'
value = @attributes[attr_name]
unless value.nil?
  if column = column_for_attribute(attr_name)
    if unserializable_attribute?(attr_name, column)
      unserialize_attribute(attr_name)
    else
      column.type_cast(value)
    end
  else
    value
  end
end
end
```



in case...

```
def _read_attribute(attr_name)
  attr_name = attr_name.to_s
  attr_name = self.class.primary_key if attr_name == 'id'
  value = @attributes[attr_name]
  unless value.nil?
    if column = column_for_attribute(attr_name)
      if unserializable_attribute?(attr_name, column)
        unserialize_attribute(attr_name)
      else
        column.type_cast(value)
      end
    else
      value
    end
  end
end
```



in case...

```
    _read_attribute(attr_name)
  attr_name = attr_name.to_s
  attr_name = self.class.primary_key if attr_name == 'id'
  value = @attributes[attr_name]
  unless value.nil?
    if column = column_for_attribute(attr_name)
      if unserializable_attribute?(attr_name, column)
        unserialize_attribute(attr_name)
      else
        column.type_cast(value)
      end
    else
      value
    end
  end
end
```

concise?

```ruby
def _read_attribute(attr_name)
  attr_name = attr_name.to_s
  attr_name = self.class.primary_key if attr_name == 'id'
  value = @attributes[attr_name]
  unless value.nil?
    if column = column_for_attribute(attr_name)
      if unserializable_attribute?(attr_name, column)
        unserialize_attribute(attr_name)
      else
        column.type_cast(value)
      end
    else
      value
    end
  end
end
```

the flow is COMPLEX

```
def _read_attribute(attr_name)
  attr_name = attr_name.to_s
  attr_name = self.class.primary_key if attr_name == 'id'
  value = @attributes[attr_name]
  unless value.nil?
    if column = column_for_attribute(attr_name)
      if unserializable_attribute?(attr_name, column)
        unserialize_attribute(attr_name)
      else
        column.type_cast(value)
      end
    else
      value
    end
  end
end
```

what flow these COMPLEX variables?

```
def _read_attribute(attr_name)
  attr_name = attr_name.to_s
  attr_name = self.class.primary_key if attr_name == 'id'
  value = @attributes[attr_name]
  unless value.nil?
    if column = column_for_attribute(attr_name)
      if unserializable_attribute?(attr_name, column)
        unserialize_attribute(attr_name)
      else
        column.type_cast(value)
      end
    else
      value
    end
  end
end
```

variable names?

Java?

variable names?

Java? C++?

variable names?

Java? C++?

variable

C?

names?

Java? C++?

variable

C?

names?

Basic?

Java? C++?

variable

C?

names?

Ruby?

Basic?

Clojure?

Java? C++?

variable

C?

names?

Ruby?

Basic?

Clojure?

Java? C++?

variable

C?

names?

Scala? Ruby?

Basic?

Clojure?

Java?

C++?

variable

This month's new language?

C?

names?

Scala?

Ruby?

Basic?

i like repeating myself

too many

ifs?

Java?

too many ifs?

Java? C++?

too many ifs?

Java?

C++?

too many

C?

ifs?

Java?

C++?

too many

C?

ifs?

Basic?

Java?

C++?

too many

C?

ifs?

Ruby?

Basic?

Java?

C++?

Clojure?

too many

C?

ifs?

Ruby?

Basic?

Clojure?

Java? C++?

too many ifs?

C?

Scala? Ruby?

Basic?

Java?

C++?

Clojure?

too many ifs?

C?

This month's new language?

Scala?

Ruby?

Basic?

new

new language

new language

new

new language

new mindset

new language

new mindset

new

new language

new mindset

new idiomatic usage

new language

new mindset

new idiomatic usage

same

new language

new mindset

new idiomatic usage

same mistakes

new language nice

new mindset

new idiomatic usage

same mistakes

new language nice

new mindset nice

new idiomatic usage

same mistakes

new language nice

new mindset nice

new idiomatic usage cute

same mistakes

new language nice

new mindset nice

new idiomatic usage cute

same mistakes what the???

```
def cached_attributes
  @cached_attributes ||= columns.select { |c| cacheable_column?(c) }.map { |col| col.name }.to_set
end
```

got it?

```
def cached_attributes
  @cached_attributes ||= columns.select { |c| cacheable_column?(c) }.map { |col| col.name }.to_set
end
```

one line = concise?

one line = pretty?

complexity,

complexity, is natural

complexity, is natural

if invisible,

complexity, is natural

if invisible, is evil

got it?

```
        def cached_attributes
          @cached_attributes ||= columns.select { |c|
cacheable_column?(c) }.map { |col| col.name }.to_set
        end
```

```ruby
def cached_attributes
  @cached_attributes ||= columns.select { |c| cacheable_column?(c) }.map { |col| col.name }.to_set
end
```

```ruby
def cached_attributes
  @cached_attributes ||= columns.select { |c| cacheable_column?(c) }.map { |col| col.name }.to_set
end
```

how frequently does flow control appear?

```ruby
def cached_attributes
  @cached_attributes ||= columns.select { |c| cacheable_column?(c) }.map { |col| col.name }.to_set
end
```

how many tests do I need?

how frequently does flow control appear?

I test

coverage = 100%

your coverage

lied

why?

```ruby
def cached_attributes
  @cached_attributes ||= columns.select { |c| cacheable_column?(c) }.map { |col| col.name }.to_set
end
```

how frequently does flow control appear?

how many ifs?

```
def cached_attributes
  @cached_attributes ||= columns.select { |c| cacheable_column?(c) }.map { |col| col.name }.to_set
end
```

how frequently does flow control appear?

how many ifs?

```
def cached_attributes
  @cached_attributes ||= columns.select { |c| cacheable_column?(c) }.map { |col| col.name }.to_set
end
```

how frequently does flow control appear?

how many ifs?

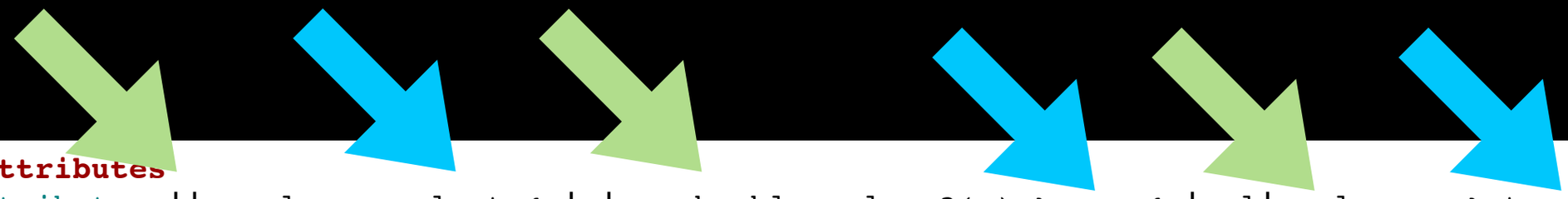
how many fors?

```
def cached_attributes
  @cached_attributes ||= columns.select { |c| cacheable_column?(c) }.map { |col| col.name }.to_set
end
```

how frequently does flow control appear?

how many ifs?

how many fors?

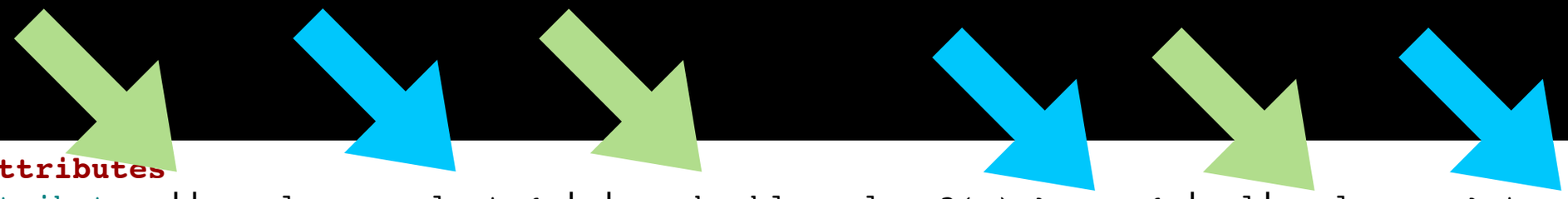


```
def cached_attributes
  @cached_attributes ||= columns.select { |c| cacheable_column?(c) }.map { |col| col.name }.to_set
end
```

how frequently does flow control appear?

how many ifs?

how many fors?



```
def cached_attributes
  @cached_attributes ||= columns.select { |c| cacheable_column?(c) }.map { |col| col.name }.to_set
end
```

how many tests do I need?

how frequently does flow control appear?

and now?

```
def cached_attributes
  @cached_attributes ||= columns.select { |c| cacheable_column?(c) }
                                .map { |col| col.name }
                                .to_set
end
```

and now?

how frequently does flow control appear?

```
def cached_attributes
  @cached_attributes ||= columns.select { |c| cacheable_column?(c) }
                                .map { |col| col.name }
                                .to_set
end
```

got it?

```
def cached_attributes
  return @cached_attributes if @cached_attributes
  cacheables = columns.select {|c|cacheable_column?(c)}
  names = cacheables.map { |col| col.name }
  @cached_attributes = names.to_set
end
```

got it?

```
def cached_attributes
  return @cached_attributes if @cached_attributes
  val cacheables = columns.select {|c|cacheable_column?(
  val names = cacheables.map { |col| col.name }
  @cached_attributes = names.to_set
end
```

2011년 11월 16일 수요일

hit enter

for your own good

due to “concision”

fewer keystrokes crapped my code

```
def cached_attributes
  @cached_attributes ||= columns.select { |c| cacheable_column?(c) }.map { |col| col.name }.to_set
end
```

it’s not about beauty

it's not about beauty

it is about not crappy code

concise = with as less words as possible, make yourself clear

concise = with as less words as possible, make yourself clear

don't mix "concise" with "crappy"

what happens if?

```
Client client = clients.lookup(15L);
client.getName();
```

what happens if?

an EJB remote request!?

```
Client client = clients.lookup(15L);
client.getName();
```

what happens if?

an EJB remote request!?

EJB 2 hell!

```
Client client = clients.lookup(15L);
client.getName();
```

what happens if?

an EJB remote request!?

EJB 2 hell!

```
Client client = clients.lookup(15L);
client.getName();
```

Joker Junior,
Meta aspect meta aspect senior developer

what happens if?

an EJB remote request!?

EJB 2 hell!

```
Client client = clients.lookup(15L);
client.getName();
```

Wanna see some magic?

Joker Junior,
Meta aspect meta aspect senior developer

what happens if?

an HTTP remote request!?

“rest” proxy hell

aka active resource

```
Client client = clients.lookup(15L);
client.getName();
```

what happens if?

```
client.save
```

```
def save
    database.save(this)
end
```

what happens if?

an HTTP remote request!?

```
client.save
```

```
def save
    database.save(this)
end
```

what happens if?

an HTTP remote request!?

ClientRule.checkUniqueEnroll is invoked!?

```
client.save
```

```
def save
    database.save(this)
end
```

what happens if?

an HTTP remote request!?

ClientRule.checkUniqueEnroll is invoked!?

The “invisible invocation(s)” pattern

```
client.save
```

```
def save
    database.save(this)
end
```

composition

composition is good

composition is good

if invisible,

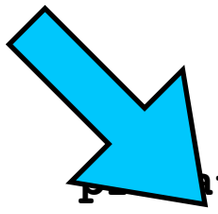
composition is good

if invisible, is evil

```
private String getIdentifierName(Class<?> cls) {
    if (!identifierNames.containsKey(cls)) {
        String name = null;

        if (cls.isAnnotationPresent(Identifier.class)) {
            Identifier identifier = (Identifier) cls.getAnnotation(Identifier.class);
            if (identifier.name() != null && !"".equals(identifier.name().trim())) {
                name = identifier.name();
            }
        }

        if (name == null) {
            name = cls.getName().substring(cls.getName().lastIndexOf('.') + 1);
        }

        identifierNames.put(cls, name);
        return name;
    }

    return identifierNames.get(cls);
}
```

9 flow breaks

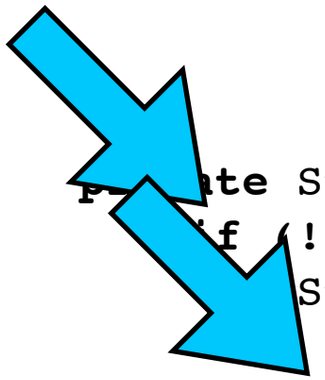
3 negations



```
    private String getIdentifierName(Class<?> cls) {
        if (!identifierNames.containsKey(cls)) {
            String name = null;

            if (cls.isAnnotationPresent(Identifier.class)) {
                Identifier identifier = (Identifier) cls.getAnnotation(Identifier.class);
                if (identifier.name() != null && !"".equals(identifier.name().trim())) {
                    name = identifier.name();
                }
            }

            if (name == null) {
                name = cls.getName().substring(cls.getName().lastIndexOf('.') + 1);
            }

            identifierNames.put(cls, name);
            return name;
        }

        return identifierNames.get(cls);
    }
```

9 flow breaks

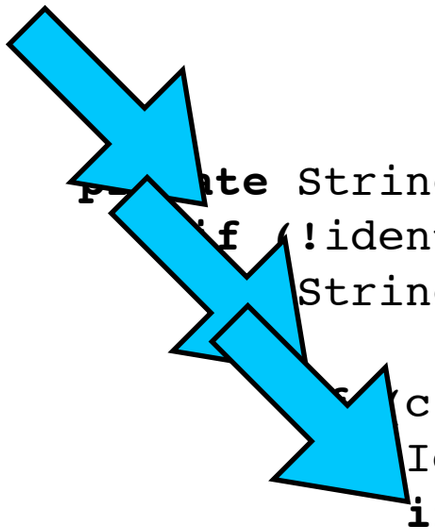
3 negations



```
    private String getIdentifierName(Class<?> cls) {
        if (!identifierNames.containsKey(cls)) {
            String name = null;

            if (cls.isAnnotationPresent(Identifier.class)) {
                Identifier identifier = (Identifier) cls.getAnnotation(Identifier.class);
                if (identifier.name() != null && !"".equals(identifier.name().trim())) {
                    name = identifier.name();
                }
            }

            if (name == null) {
                name = cls.getName().substring(cls.getName().lastIndexOf('.') + 1);
            }

            identifierNames.put(cls, name);
            return name;
        }

        return identifierNames.get(cls);
    }
```

9 flow breaks

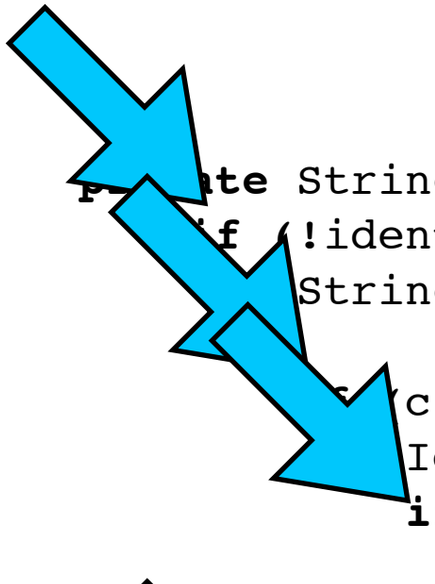
3 negations



```
private String getIdentifierName(Class<?> cls) {
    if (!identifierNames.containsKey(cls)) {
        String name = null;

        if (cls.isAnnotationPresent(Identifier.class)) {
            Identifier identifier = (Identifier) cls.getAnnotation(Identifier.class);
            if (identifier.name() != null && !"".equals(identifier.name().trim())) {
                name = identifier.name();
            }
        }

        if (name == null) {
            name = cls.getName().substring(cls.getName().lastIndexOf('.') + 1);
        }

        identifierNames.put(cls, name);
        return name;
    }

    return identifierNames.get(cls);
}
```

9 flow breaks

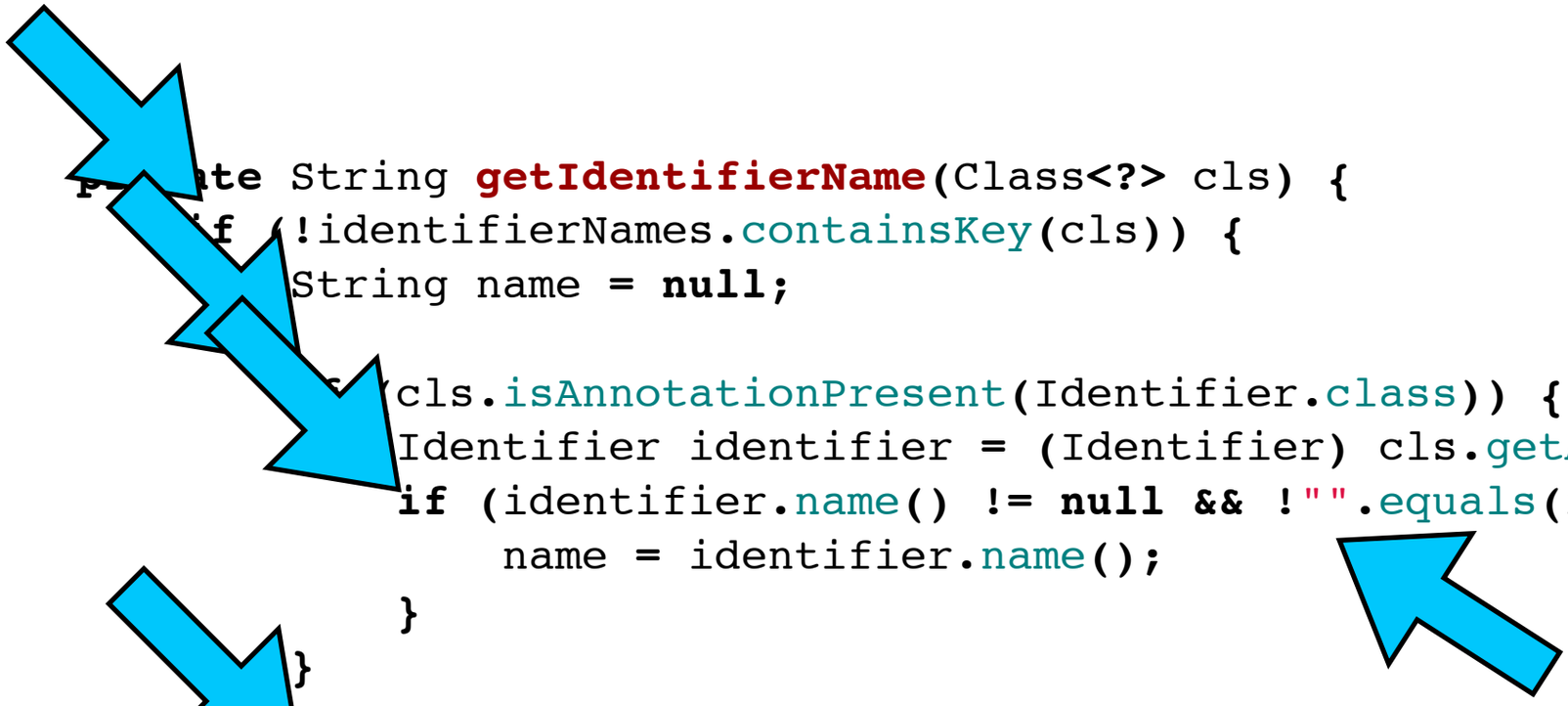
3 negations



```
private String getIdentifierName(Class<?> cls) {
    if (!identifierNames.containsKey(cls)) {
        String name = null;

        if (cls.isAnnotationPresent(Identifier.class)) {
            Identifier identifier = (Identifier) cls.getAnnotation(Identifier.class);
            if (identifier.name() != null && !"".equals(identifier.name().trim())) {
                name = identifier.name();
            }
        }

        if (name == null) {
            name = cls.getName().substring(cls.getName().lastIndexOf('.') + 1);
        }

        identifierNames.put(cls, name);
        return name;
    }

    return identifierNames.get(cls);
}
```

9 flow breaks

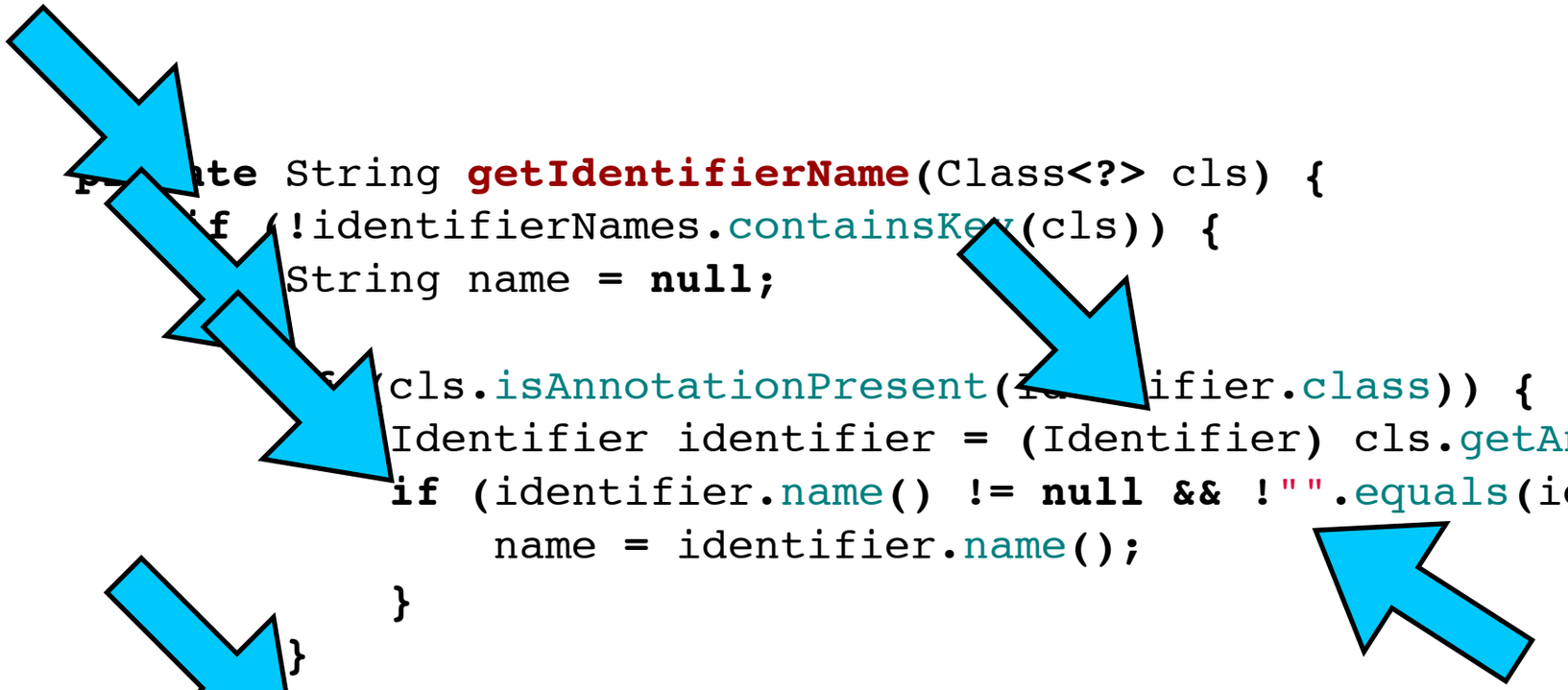
3 negations



```
private String getIdentifierName(Class<?> cls) {
    if (!identifierNames.containsKey(cls)) {
        String name = null;

        if (cls.isAnnotationPresent(Identifier.class)) {
            Identifier identifier = (Identifier) cls.getAnnotation(Identifier.class);
            if (identifier.name() != null && !"".equals(identifier.name().trim())) {
                name = identifier.name();
            }
        }

        if (name == null) {
            name = cls.getName().substring(cls.getName().lastIndexOf('.') + 1);
        }

        identifierNames.put(cls, name);
        return name;
    }

    return identifierNames.get(cls);
}
```

9 flow breaks

3 negations

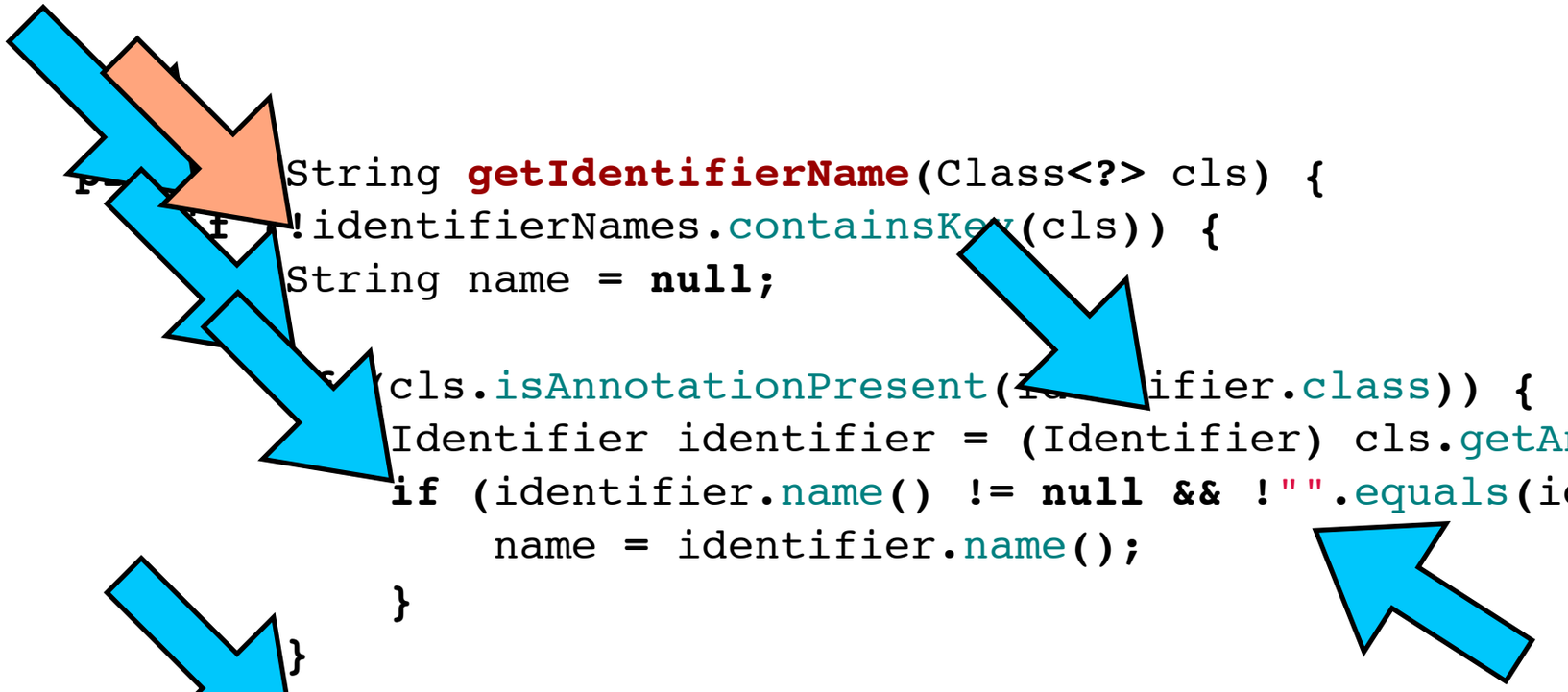


```
private String getIdentifierName(Class<?> cls) {
    if (!identifierNames.containsKey(cls)) {
        String name = null;

        if (cls.isAnnotationPresent(Identifier.class)) {
            Identifier identifier = (Identifier) cls.getAnnotation(Identifier.class);
            if (identifier.name() != null && !"".equals(identifier.name().trim())) {
                name = identifier.name();
            }
        }

        if (name == null) {
            name = cls.getName().substring(cls.getName().lastIndexOf('.') + 1);
        }

        identifierNames.put(cls, name);
        return name;
    }

    return identifierNames.get(cls);
}
```

9 flow breaks

3 negations

```
    String getIdentifierName(Class<?> cls) {
    if (!identifierNames.containsKey(cls)) {
        String name = null;

        if (cls.isAnnotationPresent(Identifier.class)) {
            Identifier identifier = (Identifier) cls.getAnnotation(Identifier.class);
            if (identifier.name() != null && !"".equals(identifier.name().trim())) {
                name = identifier.name();
            }
        }

        if (name == null) {
            name = cls.getName().substring(cls.getName().lastIndexOf('.') + 1);
        }

        identifierNames.put(cls, name);
        return name;
    }

    return identifierNames.get(cls);
}
```



9 flow breaks

3 negations

```
    String getIdentifierName(Class<?> cls) {
    if (!identifierNames.containsKey(cls)) {
        String name = null;

        if (cls.isAnnotationPresent(Identifier.class)) {
            Identifier identifier = (Identifier) cls.getAnnotation(Identifier.class);
            if (identifier.name() != null && !"".equals(identifier.name().trim())) {
                name = identifier.name();
            }
        }

        if (name == null) {
            name = cls.getName().substring(cls.getName().lastIndexOf('.') + 1);
        }

        identifierNames.put(cls, name);
        return name;
    }

    return identifierNames.get(cls);
}
```

9 flow breaks

3 negations

```
    String getIdentifierName(Class<?> cls) {
    if (!identifierNames.containsKey(cls)) {
        String name = null;

        if (cls.isAnnotationPresent(Identifier.class)) {
            Identifier identifier = (Identifier) cls.getAnnotation(Identifier.class);
            if (identifier.name() != null && !"".equals(identifier.name().trim())) {
                name = identifier.name();
            }
        }

        if (name == null) {
            name = cls.getName().substring(cls.getName().lastIndexOf('.') + 1);
        }

        identifierNames.put(cls, name);
        return name;
    }

    return identifierNames.get(cls);
}
```

9 flow breaks

3 negations

```
private String getIdentifierName(Class<?> cls) {
    if (!identifierNames.containsKey(cls)) {
        String name = null;

        if (cls.isAnnotationPresent(Identifier.class)) {
            Identifier identifier = (Identifier) cls.getAnnotation(Identifier.class);
            if (identifier.name() != null && !"".equals(identifier.name().trim())) {
                name = identifier.name();
            }
        }

        if (name == null) {
            name = cls.getName().substring(cls.getName().lastIndexOf('.') + 1);
        }

        identifierNames.put(cls, name);
        return name;
    }

    return identifierNames.get(cls);
}
```

```
private String getIdentifierName(Class<?> cls) {
    if (identifierNames.containsKey(cls)) {
        return identifierNames.get(cls);
    }

    String name = null;

    if (cls.isAnnotationPresent(Identifier.class)) {
        Identifier identifier = (Identifier) cls.getAnnotation(Identifier.class);
        if (identifier.name() != null && !"".equals(identifier.name().trim())) {
            name = identifier.name();
        }
    }

    if (name == null) {
        name = cls.getName().substring(cls.getName().lastIndexOf('.') + 1);
    }

    identifierNames.put(cls, name);
    return name;

}
```

```java
private String getIdentifierName(Class<?> cls) {
    if (identifierNames.containsKey(cls)) {
        return identifierNames.get(cls);
    }

    String name = null;

    Identifier identifier = (Identifier) cls.getAnnotation(Identifier.class);
    if (identifier != null && identifier.name() != null && !"".equals(identifier.name().trim()
        name = identifier.name();
    }

    if (name == null) {
        name = cls.getName().substring(cls.getName().lastIndexOf('.') + 1);
    }

    identifierNames.put(cls, name);
    return name;

}
```

```
private String getIdentifierName(Class<?> cls) {
    if (identifierNames.containsKey(cls)) {
        return identifierNames.get(cls);
    }

    String name = null;

    Identifier identifier = (Identifier) cls.getAnnotation(Identifier.class);
    if (identifier != null && identifier.name() != null && !"".equals(identifier.name().trim()
        name = identifier.name();
    }

    if (name == null) {
        name = cls.getName().substring(cls.getName().lastIndexOf('.') + 1);
    }

    return cache(cls, name);

}


private String cache(Class<?> cls, String value) {
  identifierNames.put(cls, name);
  return value;
}
```

```
private String getIdentifierName(Class<?> cls) {
    if (identifierNames.containsKey(cls)) {
        return identifierNames.get(cls);
    }

    Identifier identifier = (Identifier) cls.getAnnotation(Identifier.class);
    if (identifier == null || identifier.name() == null || "".equals(identifier.name().trim())
        return cache(cls, cls.getName().substring(cls.getName().lastIndexOf('.') + 1));
    }

    return cache(cls, identifier.name());

}

private String cache(Class<?> cls, String value) {
  identifierNames.put(cls, name);
  return value;
}
```

```
private String getIdentifierName(Class<?> cls) {
    if (identifierNames.containsKey(cls)) {
        return identifierNames.get(cls);
    }

    Identifier identifier = (Identifier) cls.getAnnotation(Identifier.class);
    if (identifier == null || identifier.name() == null || "".equals(identifier.name().trim())
        return cache(cls, nameFor(cls));
    }

    return cache(cls, identifier.name());

}

private String cache(Class<?> cls, String value) {
  identifierNames.put(cls, name);
  return value;
}

private String nameFor(Class<?> cls) {
  return cls.getName().substring(cls.getName().lastIndexOf('.') + 1);;
}
```

```
private String getIdentifierName(Class<?> cls) {
    if (identifierNames.containsKey(cls)) {
        return identifierNames.get(cls);
    }

    Identifier identifier = (Identifier) cls.getAnnotation(Identifier.class);
    if (identifier == null || nullOrEmpty(identifier.name())) {
        return cache(cls, nameFor(cls));
    }

    return cache(cls, identifier.name());

}
```

3 flow breaks

0 negations

```
private static boolean nullOrEmpty(String value) {
  return value == null || "".equals(value.trim());
}

private String cache(Class<?> cls, String value) {
  identifierNames.put(cls, name);
  return value;
}

private String nameFor(Class<?> cls) {
  return cls.getName().substring(cls.getName().lastIndexOf('.') + 1);;
}
```

2 flow breaks

0 negations

```
    ate String getIdentifierName(Class<?> cls) {
    if (identifierNames.containsKey(cls)) {
        return identifierNames.get(cls);


    Ide   fier identifier = (Ide   fier) cls.getAnnotation(Identifier.class);
    if (identifier == null || nullOrEmpty(identifier.name())) {
        return cache(cls, nameFor(cls));
    }

    return cache(cls, identifier.name());

}

private sta   boolean nullOrEmpty   ing value) {
  return value == null || "".equals(   lue.trim());
}

private String cache(Class<?> cls, String value) {
  identifierNames.put(cls, name);
  return value;
}

private String nameFor(Class<?> cls) {
  return cls.getName().substring(cls.getName().lastIndexOf('.') + 1);;
}
```

3 flow breaks
0 negations

2 flow breaks
0 negations

it is easy

it is easy to start with a mess

it is easy to start with a mess

it is hard

it is easy to start with a mess

it is hard to start right

it is easy to start with a mess

it is hard to start right

it is risky

it is easy to start with a mess

it is hard to start right

it is risky to refactor later

it is easy to start with a mess

it is hard to start right

it is risky to refactor later

it is easy

it is easy to start with a mess

it is hard to start right

it is risky to refactor later

it is easy to refactor soon

it is easy to start with a mess

it is hard to start right

it is risky to refactor later

it is easy to refactor soon

After 10 years. Which ones will you pick?

it is easy to start with a mess

it is hard to start right

it is risky to refactor later

it is easy to refactor soon

After 10 years. Which ones will you pick?

After 10 years. Are you still trying to start right?

it is easy to start with a mess

it is hard to start right

it is risky to refactor later

it is easy to refactor soon

After 10 years. Which ones will you pick?

After 10 years. Are you still trying to start right?

After 10 years. Are you still leaving for later?

“if if if if if if” hurts

this one hurts?

select * from Contacts where a=b or (c=d && e=f)
order by first_field limit 2

this one hurts?

select * from Contacts where a=b or (c=d && e=f)
order by first_field limit 2

Maurício Aniche

this one hurts?

select * from Contacts where a=b or (c=d && e=f)
order by first_field limit 2

How would you test it?

Maurício Aniche

this one hurts?

select * from Contacts where a=b or (c=d && e=f)
order by first_field limit 2

How would you test it?

mock.assert(_.select(“select from ...”))?

Maurício Aniche

this one hurts?

select * from Contacts where a=b or (c=d && e=f)
order by first_field limit 2

How would you test it?

mock.assert(_.select(“select from ...”))?

I test?

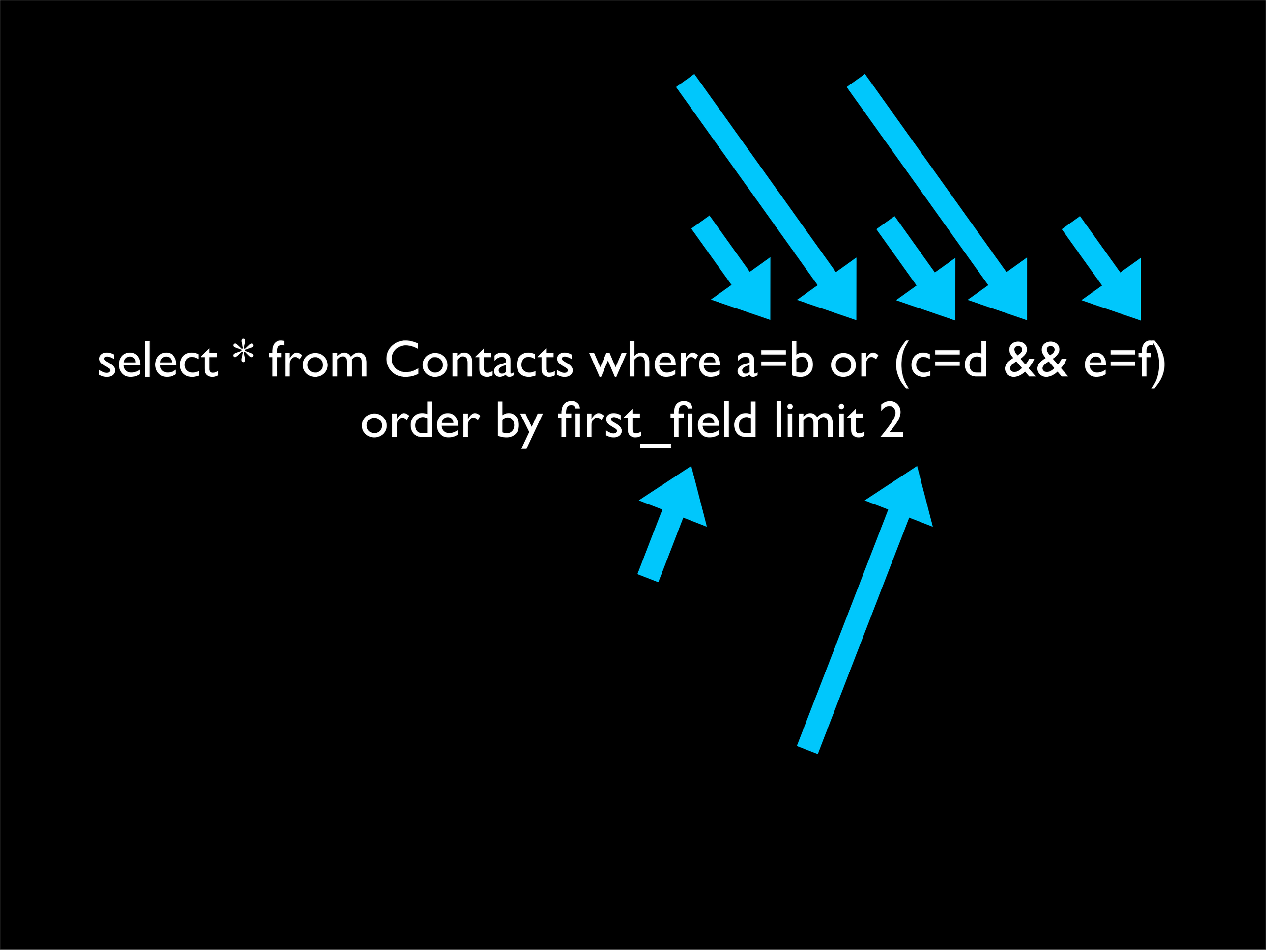
Maurício Aniche

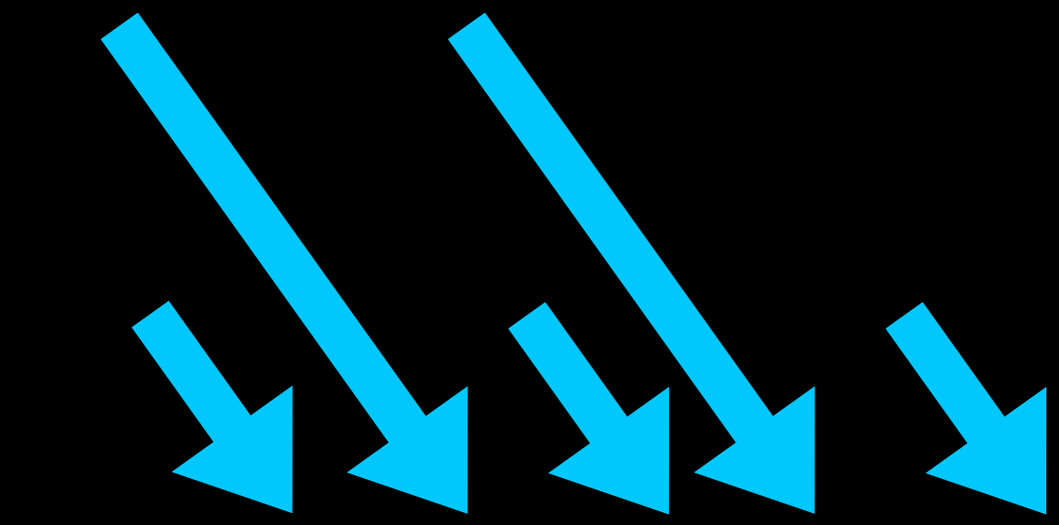
your coverage

lies

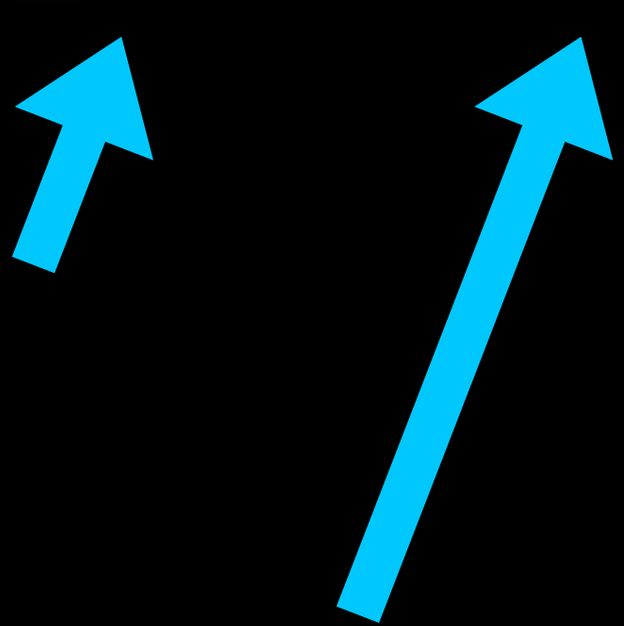
why?

```
select * from Contacts where a=b or (c=d && e=f)
order by first_field limit 2
```

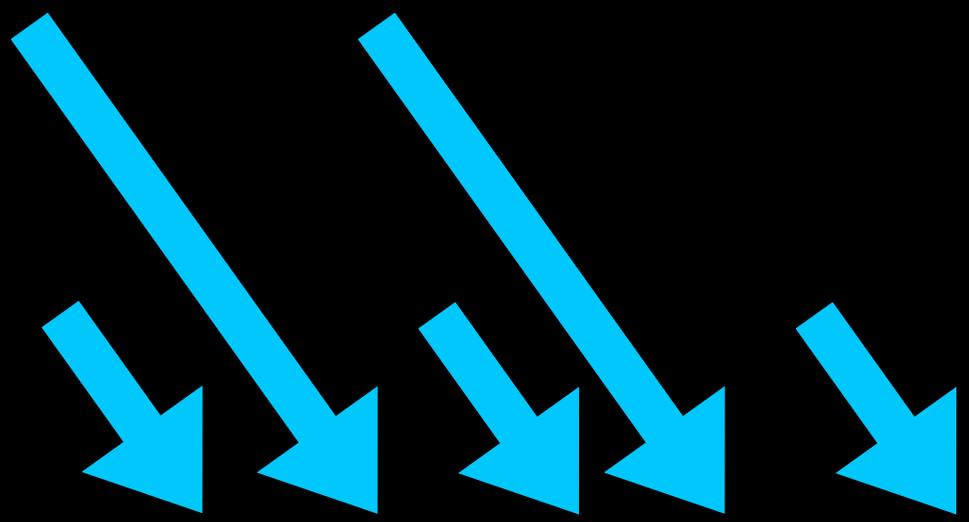

select * from Contacts where a=b or (c=d && e=f)
order by first_field limit 2



select * from Contacts where a=b or (c=d && e=f)
order by first_field limit 2



Martin Fowler



```
select * from Contacts where a=b or (c=d && e=f)
order by first_field limit 2
```

sql is also a language...

Martin Fowler

less than 5 tests?

but coverage says 100%...

coverage tools lie

coverage 100% would be

128 tests?

there is no if without a if

there is no loop without a loop

only if you hide it!

hiding complexity makes other
not think it is there
it's worse!

it is still there...
but noone knows!!!

complexity: do not hide it

instead, make it easier to understand it

08 101

2011년 11월 16일 수요일

if there is something you need
to execute every time
you create a new Client?

if there is something you need

to execute every time

you create a new Client?

if there is something you need

to execute every time

you create a new Client?

declare your dependencies
and use your
CONSTRUCTOR

declare your dependencies and use your CONSTRUCTOR

but what does Rails, Javabeans, Hibernate etc say about constructors and members?

java (the javabeans way)

dumb constructor

getter+setter

rails (the ruby way)

dumb constructor

attr_accessor

the scala way

dumb constructor

val

java (the javabeans way)

dumb constructor

public variable

rails (the ruby way)

dumb constructor

public variable

the scala way

dumb constructor

public variable

standard OO in the market is
the "object orgy" pattern.

anyone touches anyone's else
private (aka public) parts

is it true?

Spring: CTRL+F static

Rails: YourActiveRecord.new.methods.size

Any language:
client.projects[0].product.detail.price

orgy

positioning

code

a new scala project

vraptor + hibernate

Can this user mark this post as solved?

```
def canSolve(post: Post) =
    post.discussion.solvable &&
      post.discussion.author.equals(this)
```

If post or discussion changes, I will break.

Can this user mark this post as solved?

```
def canSolve(post: Post) =
    post.discussion.solvable &&
      post.discussion.author.equals(this)
```

If post or discussion changes, I will break.

Can this user mark this post as solved?

```
def canSolve(post: Post) =
    post.discussion.solvable &&
      post.discussion.author.equals(this)
```

If post or discussion changes, I will break.

Can this user mark this post as solved?

```
def canSolve(post: Post) =
    post.discussion.solvable &&
      post.discussion.author.equals(this)
```

If post or discussion changes, I will break.

Can this user mark this post as solved?

```
def canSolve(post: Post) =
    post.discussion.solvable &&
      post.discussion.author.equals(this)
```

If post or discussion changes, I will break.

it is basic, but how often

do we see it?

find the

right place

for your code

```
def canSolve(post: Post) =
    post.discussion.solvable &&
      post.discussion.author.equals(this)
```

post post post

Ah! Because you had public vars!

```
def canSolve(post: Post) =
    post.discussion.solvable &&
      post.discussion.author.equals(this)
```

post post post

Ah! Because you had public vars!

```
def canBeSolvedBy(user: User) =
  discussion.solvable &&
    discussion.author.equals(user)
```

```
def canSolve(post: Post) =
    post.discussion.solvable &&
      post.discussion.author.equals(this)
```

post post post

Ah! Because you had public vars!

```
def canBeSolvedBy(user: User) =
  discussion.solvable &&
    discussion.author.equals(user)
```

```
def canSolve(post: Post) =
    post.discussion.solvable &&
      post.discussion.author.equals(this)
```

post post post

Ah! Because you had public vars!

```
def canBeSolvedBy(user: User) =
    discussion.solvable &&
      discussion.author.equals(user)
```

in the right place

```
def canBeSolvedBy(user: User) =
  solvable && author.equals(user)
```

I load a post and check if it can be marked as solved.

encapsulation++

demeter++

in the right place

```
def canBeSolvedBy(user: User) =
  solvable && author.equals(user)
```

I load a post and check if it can be marked as solved.

encapsulation++

demeter++

model RULES,

do not model models

+ UP

insufficient
architectural aspects

==>

improve it

otherwise

1. implementation exists, the rest is an interpretation

2. code is complex

2. code is complex
make it good
no worry for
extra cuteness

3. hiding complexity is not simplifying

4. if it is hard to test,
it is hard to use

5. let your devs learn and improve

6. when changing
languages
triple check
they kept
the principles

7. improve

7. refactor

7. refactor all the time

7. refactor
all the time
for the better
not for the prettier

remember

pair-programming
brown bag refactoring
code review

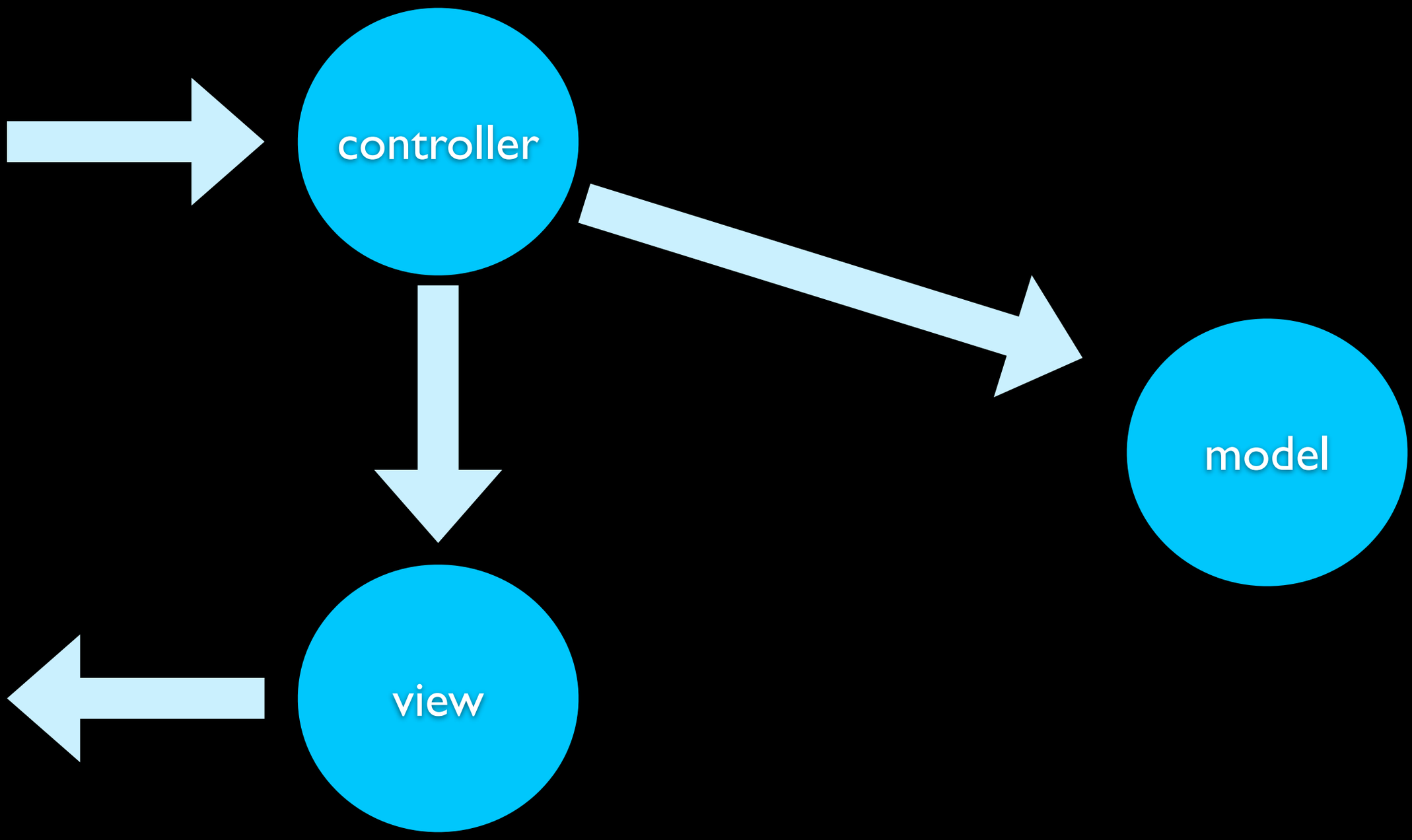
@guilhermecaelum

guilherme.silveira@caelum.com.br

http://bit.ly/quality_trial

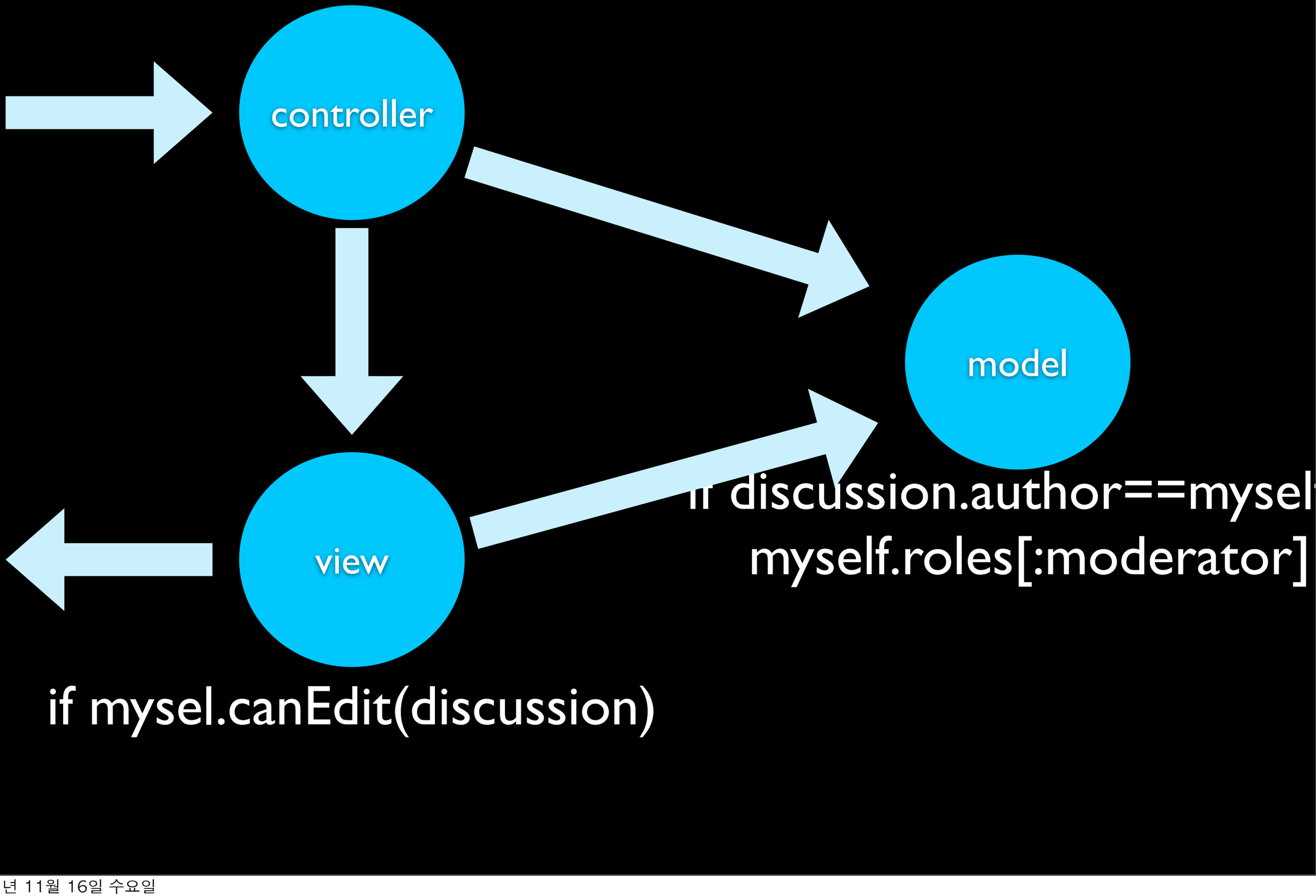
thanks

presentation code != domain code




```
if discussion.author==myself ||
    myself.roles[:moderator]
```

presentation code != domain code



find the

right place

for your code

even a template is code.

treat it as code.