Introduction to Ruby, MVC, and the Rails Framework

Professor Larry Heimann
Application Design & Development
Information Systems Program

Philosophy of Ruby

“For me, the purpose of life is, at least partly, to have joy. Programmers often feel joy when they can concentrate on the creative side of programming, so Ruby is designed to make programmers happy.”

— Yukihiro Matsumoto

Three Principles

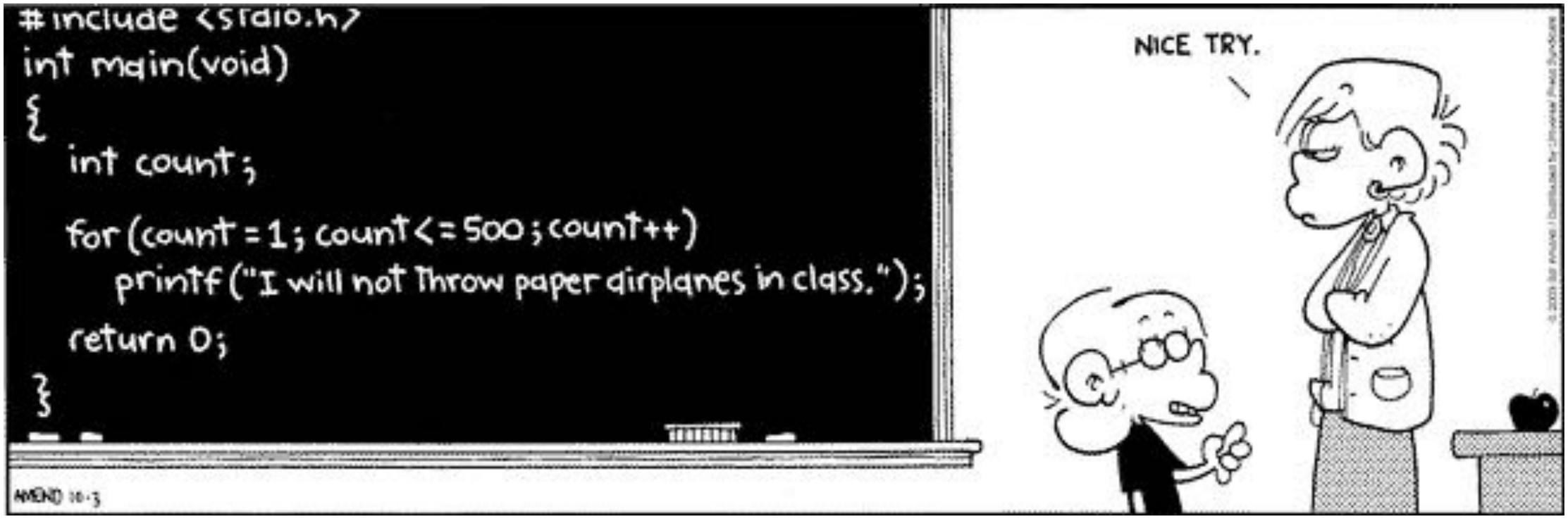
1. **Conciseness**—Writing code in Ruby should involve the minimum amount of commands necessary. Code should be terse but also understandable.

2. **Consistency**—Ruby coding should follow common conventions that make coding intuitive and unambiguous.

3. **Flexibility**—There is no one right way. You should be able to pick the best approach for your needs and be able to even modify the base classes if necessary.

   These three together lead to an important concept in Ruby —
   the principle of least surprise.

Comic of the Day...



The Ruby Way

```ruby
500.times { puts "I will not throw paper airplanes" }

(1..500).each { |i| puts "I will not throw paper airplanes" }

for i in (1..500) do
  puts "#{i}. I will not throw paper airplanes"
end
```

Everything is an object

Looking at Strings, we see:

```ruby
phrase = "i AM arthur, king of the britons"

puts phrase.class             # >> String
puts phrase.length            # >> 32
puts phrase.capitalize        # >> I am arthur, king of the britons
puts phrase.upcase            # >> I AM ARTHUR, KING OF THE BRITONS
puts phrase.downcase          # >> i am arthur, king of the britons
puts phrase.reverse           # >> snotirb eht fo gnik ,ruhtra MA i
puts phrase.upcase.reverse    # >> SNOTIRB EHT FO GNIK ,RUHTRA MA I
puts phrase.split             # >> i
                              # >> AM
                              # >> arthur,
                              # >> king
                              # >> of
                              # >> the
                              # >> britons
puts phrase.split('a')        # >> i AM
                              # >> rthur, king of the britons
puts phrase.index('a')        # >> 5
puts phrase[5..10]            # >> arthur
puts phrase.capwords          # =>
# ~> -:14: undefined method `capwords' for "i AM arthur, king of the britons":String (NoMethodError)
```

Revising the String class

```ruby
class String
  def capwords
    words = self.split
    revised = %w[]
    words.each do |word|
      revised << word.capitalize
    end
    final =  revised.join(" ")
  end
end

phrase = "i AM arthur, king of the britons"
phrase.capwords  # => "I Am Arthur, King Of The Britons"
```

Destructive and Predicate methods

```
str = "fred"
str.capitalize  # => "Fred"
puts str        # >> fred
str.capitalize! # => "Fred"
puts str        # >> Fred
str.reverse     # => "derF"
puts str        # >> Fred
str.reverse!    # => "derF"
puts str        # >> derF

str.include?('ed') # => true
```

Architecting Software

- Needs to be:
    - understandable
    - extensible
- Many different architecture patterns exist
- Model-View-Controller (MVC) one of the most popular

MVC is like ...

Model: Taking Care of Business

View: Looking Good

View: Partials

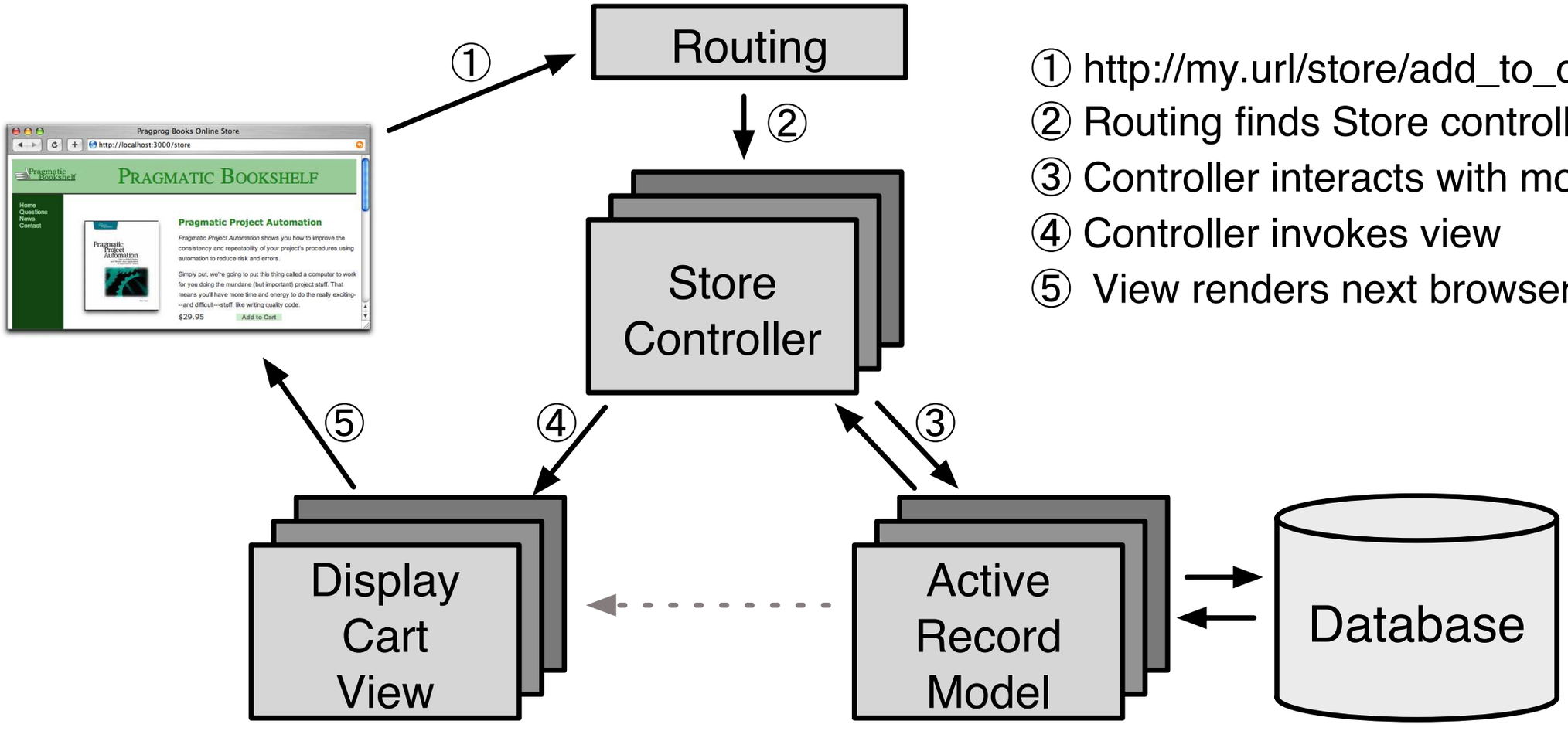
Controller: Holding It All Together

Controllers: Too Fat To Be Useful

Controllers: Variations

Controller: Traffic Cop

MVC as used in Rails



① http://my.url/store/add_to_cart/123
② Routing finds Store controller
③ Controller interacts with model
④ Controller invokes view
⑤ View renders next browser screen

To know the Model-View-Controller,

you must *be* the

Models, Views and Controllers...