高级软件工程

第五周（Oct. 11）

主讲：罗铁坚

助教：周文璋 、俞永生、姚敏

上课： 周一、三 上午 10:30—12:10

答疑：周一、三 下午 2:30—3:30   (学园2-485)

联系：tjluo@ucas.ac.cn   69671829

2021高级软件工程

群号：544158863

扫一扫二维码，加入群聊。

QQ

提纲

1. 为什么1980-2000年的计算机人工智能博士论文关于手写数字识别率都不上80%?

几种不同观点

(1) 计算机算力不够

(2) 样本数量和质量

(3) 模型性能

(4) 图片处理计数不够强大

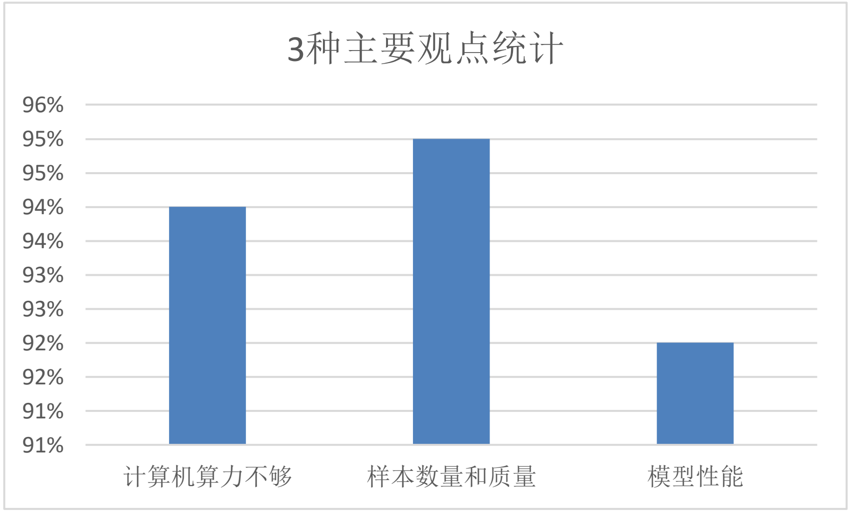
(5) 没有成熟的训练框架

(6) 机器学习正处于起步阶段，经验不足

(7) 缺少高性能框架，自己写的性能有缺陷

(8) 预处理和参数设置不当

(9) 人为归纳特征无法覆盖全部特征

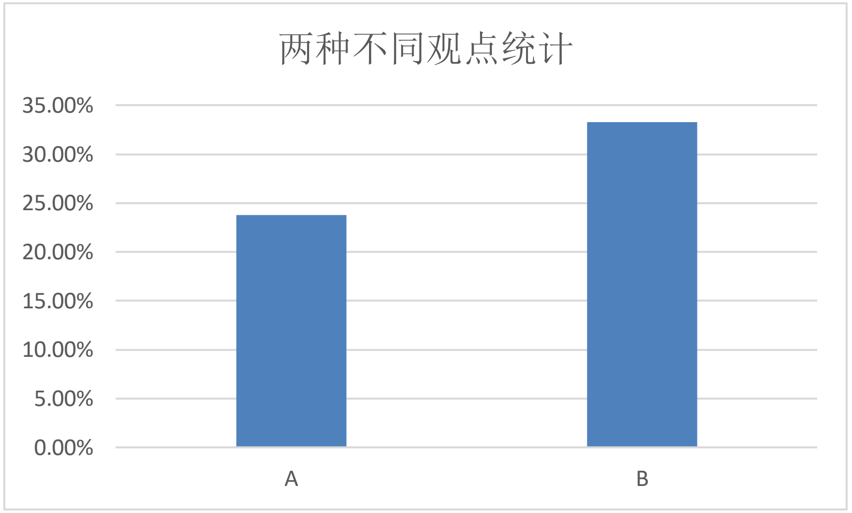


中国科学院大学

2. 人总结规律用代码写出程序的方法与用数据集训练的程序的共同点和不同点

相同点	不同点
对数据集特点的总结	人善于发现直观，计算机善于发现潜在规律
都有模型数据	人用代码驱动模型，机器用数据驱动模型
依赖数据，计算机，开发工具，需要评价指标	人设计方法以来人的经验，计算机以来数据和训练方法
提取特征，寻找特征，分类标签的映射规律	人找的规律具有很强主观性，计算机依赖数据更合理，但耗时易过拟合
通过不断迭代演化生成function	人总结规律集中设计编码和规则，数据集训练主要设计模型流程
都是基于数据集进行训练	人工代码效率更高，数据训练逻辑性和解释性更低
解决某个问题需要数据支持	人总结规律是透明，具体可解释；机器的规律是不透明

2. 人总结规律用代码写出程序的方法与用数据集训练的程序的共同点和不同点



A: 人设计-透明可解释性；机器学习-解释性低

B: 人设计-主观性；机器学习-逻辑性强更合理

3. 你认为要利用这个模型实现一个有价值的软件还需要做哪些工作?

几种不同观点

(1) 提高模型准确性

(2) 具体应用场景

(3) 前段交互，反馈，扩展功能

(4) 考虑平台依赖性和兼容性

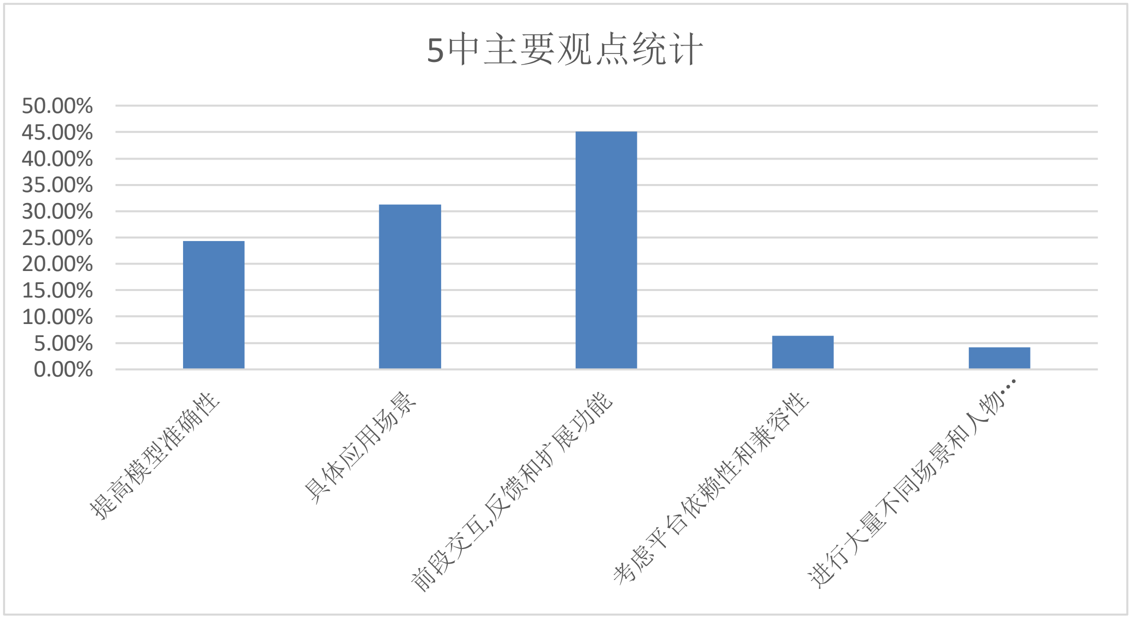
(5) 进行大量不同场景和人物测试

(6) 在合适平台发布

(7) 找到有需求的用户群体

(8) 需要配套硬件

(9) 进行市场调研和需求调研



提纲

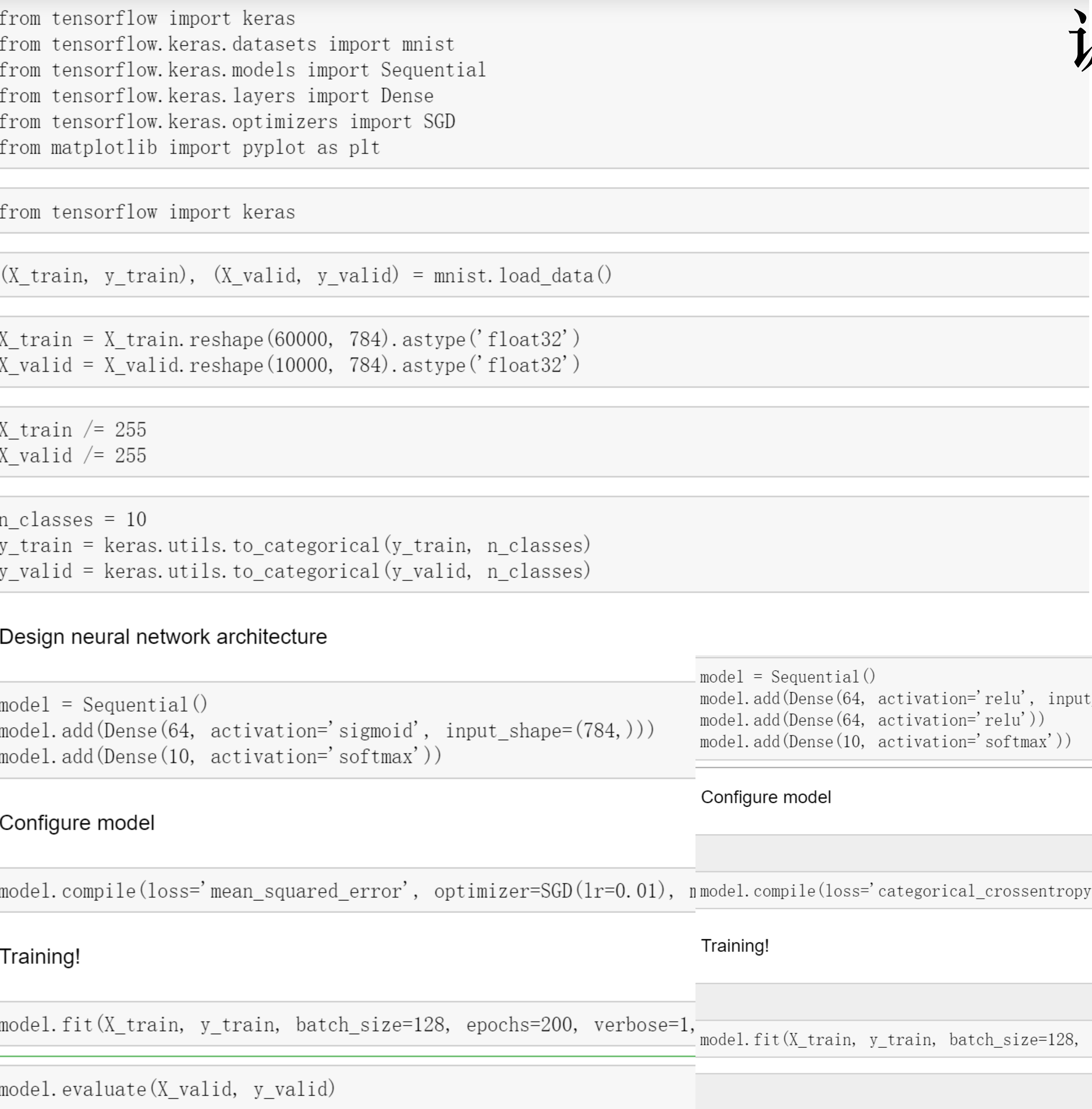
1、课堂练习情况

2、回顾2种软件设计方式

3、软件三大定律

4、BDD案例分析

训练OCR的代码



```
from tensorflow import keras
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import SGD
from matplotlib import pyplot as plt
```

```
from tensorflow import keras
```

```
(X_train, y_train), (X_valid, y_valid) = mnist.load_data()
```

```
X_train = X_train.reshape(60000, 784).astype('float32')
X_valid = X_valid.reshape(10000, 784).astype('float32')
```

```
X_train /= 255
X_valid /= 255
```

```
n_classes = 10
y_train = keras.utils.to_categorical(y_train, n_classes)
y_valid = keras.utils.to_categorical(y_valid, n_classes)
```

Design neural network architecture

```
model = Sequential()
model.add(Dense(64, activation='sigmoid', input_shape=(784,)))
model.add(Dense(10, activation='softmax'))
```

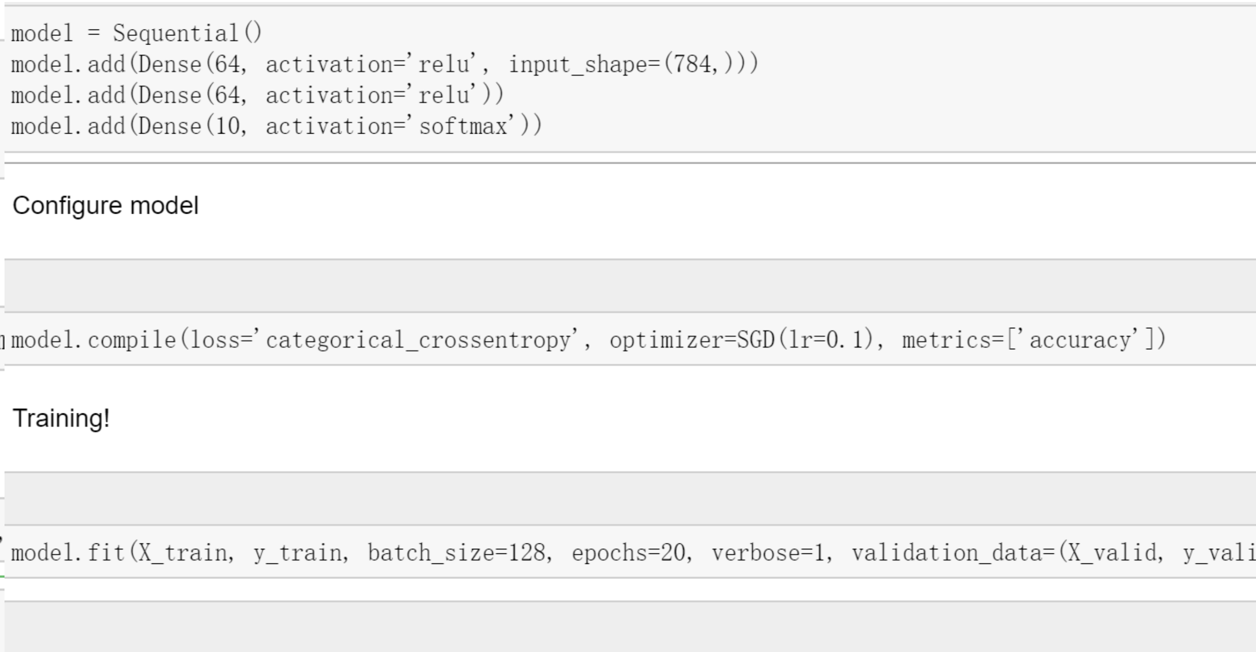
Configure model

```
model.compile(loss='mean_squared_error', optimizer=SGD(lr=0.01), n
```

Training!

```
model.fit(X_train, y_train, batch_size=128, epochs=200, verbose=1,
```

```
model.evaluate(X_valid, y_valid)
```



```
model = Sequential()
model.add(Dense(64, activation='relu', input_shape=(784,)))
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

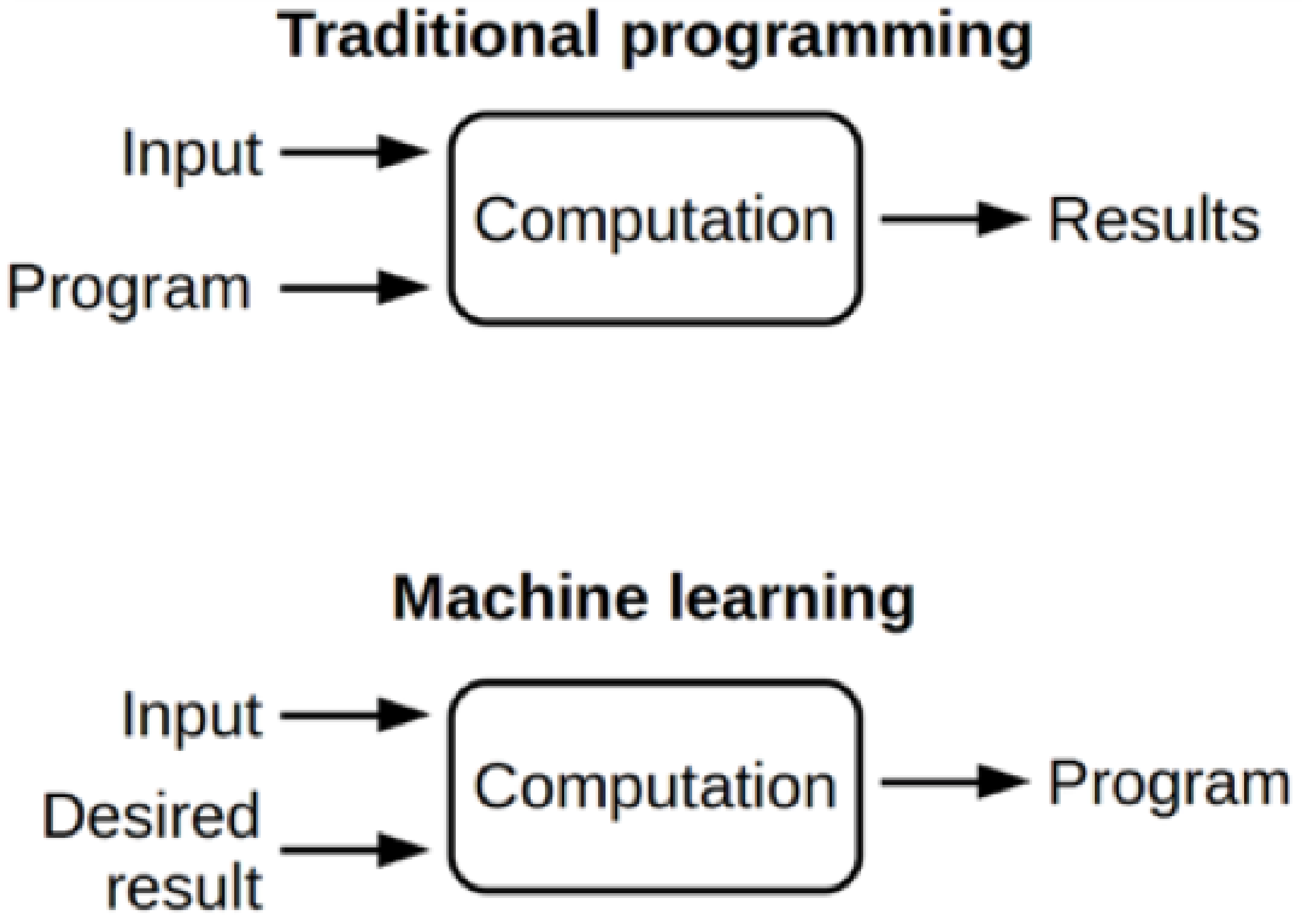
Configure model

```
model.compile(loss='categorical_crossentropy', optimizer=SGD(lr=0.1), metrics=['accuracy'])
```

Training!

```
model.fit(X_train, y_train, batch_size=128, epochs=20, verbose=1, validation_data=(X_valid, y_vali
```

软件工程 VS 机器学习



Software engineering is the art of automating a task by writing rules for a computer to follow.

Machine learning goes a step further: it automates the task of writing the rules.

Traditionally programmers automate tasks by writing programs.
In machine learning, a computer finds a program that fits to data.

两种途径设计实现软件

1. 第一种途径是：人类发现问题，提出解决方案（构思模型），然后设计代码（程序）来让计算机执行来解决问题。案例1 操作系统（Linux)、案例2 应用框架 (Ruby on Rail）、案例5 软件版本管理与持续集成平台(GitHub)是这种软件设计方式的例子。
2. 第二种途径是，人类发现问题，但没法构思出有效的解决方案（模型），人类设计一些学习程序，让计算机执行学习程序，根据给出的样本数据，计算机训练出解决方案（模型），然后集成到计算机中，即把人编制的代码和机器产生的代码结合一起运行来解决问题。案例3 玩魔方软件、案例4 池塘大战软件、AlphaGo, AlphaFold等是这种软件设计方式的例子。

用数据训练软件的的5个步骤

1. 提出业务领域的拟回答的问题。如上例就是分类问题(0-9)。
2. 收集历史上对回答这个问题的相关数据；
3. 研究对应于上面的拟做为输入的数据，让计算机推断出的可观察到的相应输出数据是什么？（这个叫数据工程，或数据清理，实际上是相当有挑战，而且也是最容易使训练出来的模型产生“偏见”的关键地方）
4. 选择机器学习中的训练算法，编写从数据加载，到得到模型的程序。
5. 把训练模型的程序集成到常规软件的应用界面，最后上线应用。

观察这必须的五步，你是否得出结论：上面的过程人工成分是否多，而且也是不可或缺，甚至是关键的。

智能软件的挑战问题

由于目前人类对想象力和提问能力还没有完全理解，如何交给机器解决也没有想明白。

因此，由计算机单独完成上面的五项任务，是有可能，但需要创新理论和方法。

我衷心希望看这本讲义的读者来尝试挑战这个难题，即教计算机单独完成五项任务。

提纲

1、课堂练习情况

2、回顾2种软件设计方式

3、软件三大定律

4、BDD案例分析

软件第一定律

任何有生命力的软件要满足一下的充分必要条件:

1.必须具备拓展人类解决问题的能力，而且找到衡量软件外部可观察行为能力的评价指标;

2.软件系统能力是通过其外部可观察及可测试验证来界定的，而且任何软件内部结构或数据的设计与实现必须支持软件系统能力的升级变化要求。

3.软件系统能力一定与某些具体的业务问题相关，而且要设计一种创新运营模式，让软件能力产生价值，从而支持软件能力的不断持续升级。

软件第二定律

任何实现用户价值的功能一定通过与运行系统交互完成，而每个具体交互有一个明确目的和操作；即每个功能都有一个具体例子来验证这个功能的实例，具体表现为UI或API的五个要素：时间（Z）、角色（Y）、X（作什么）、W（对谁，系统）、V（交换什么数据）等。

软件第三定律

所有的软件技术进步都要经受不断设计出更合理的软件内部架构及其支撑软件的持续升级的检验。

贯通“需求、设计、代码”到“运行系统”

如何设计一种“语言”来表达需求，而且能让领域专家和软件设计人员都能理解，并为设计软件架构、代码和测试提供依据，最好是也能让计算机可以自动处理。

软件开发的创新或创造表现在

1、发现解决哪些领域问题，及其价值所在；

2、创作出能用计算机解决领域问题的用例（用户故事、功能描述等）和场景；

提纲

1、课堂练习情况

2、回顾2种软件设计方式

3、软件三大定律

4、BDD案例分析

Getting Started with Rails

This guide covers getting up and running with Ruby on Rails.

After reading this guide, you will know:

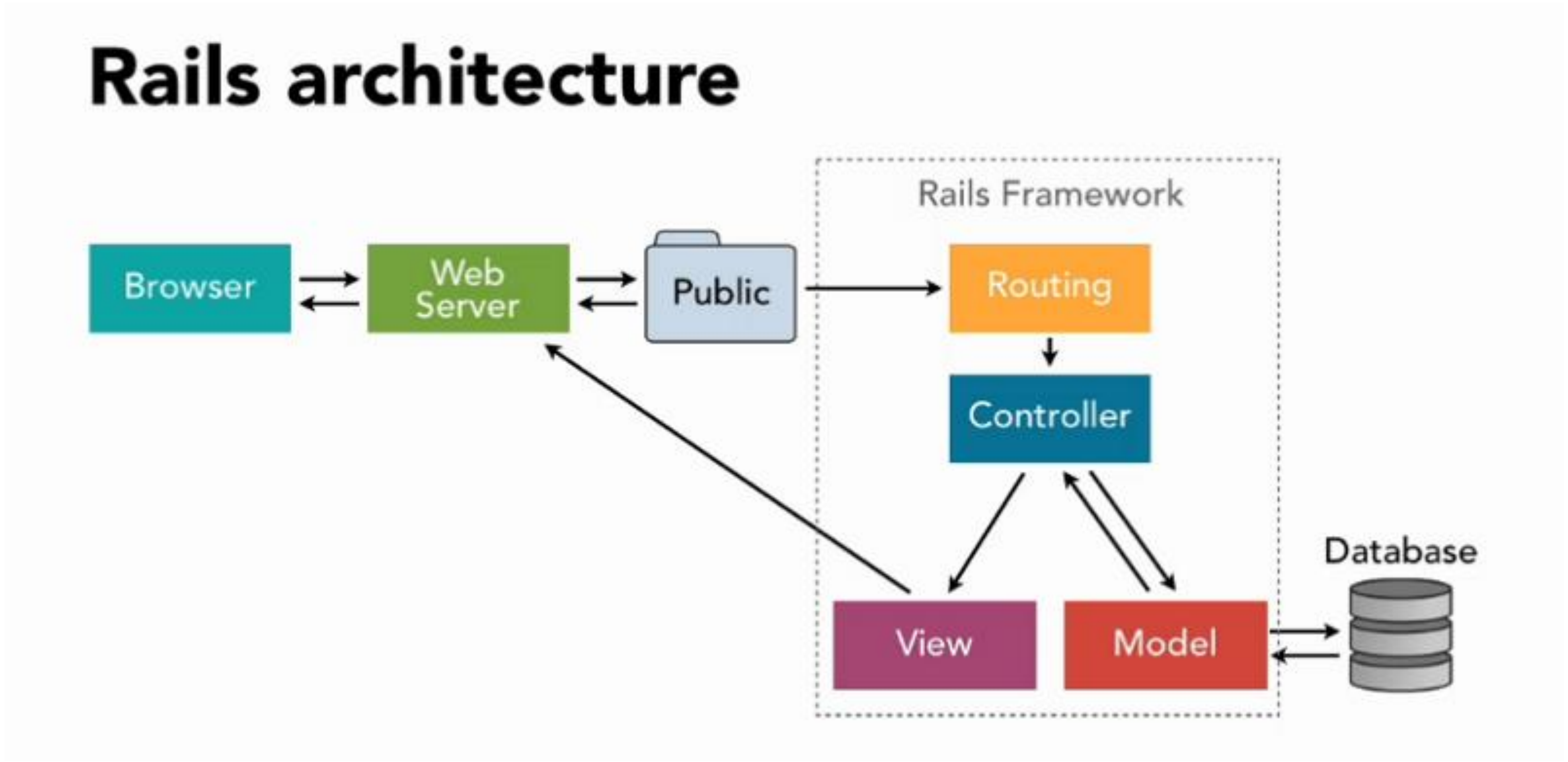
- ✓ **How to install Rails, create a new Rails application, and connect your application to a database.**
- ✓ **The general layout of a Rails application.**
- ✓ **The basic principles of MVC (Model, View, Controller) and RESTful design.**
- ✓ **How to quickly generate the starting pieces of a Rails application.**

用Django 设计开发Blog 应用需要20多个包

用Ruby on Rails 设计开发Blog，需要多少Gems?

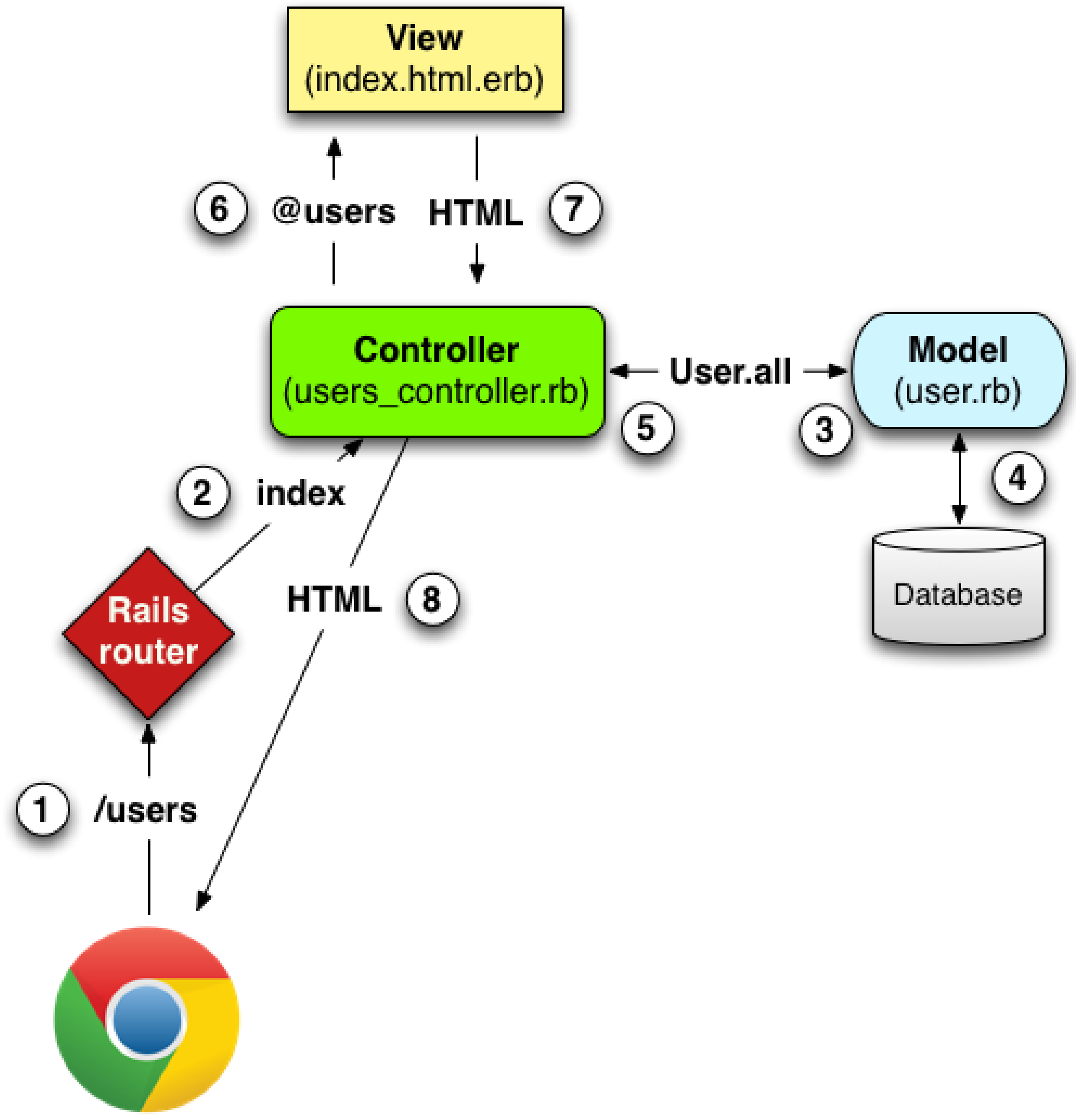
```
certifi==2019.6.16
chardet==3.0.4
defusedxml==0.6.0
Django==2.2.13
django-allauth==0.39.1
django-ckeditor==5.6.1
django-js-asset==1.2.2
django-model-utils==3.1.2
django-mptt==0.10.0
django-notifications-hq==1.5.0
django-password-reset==2.0
django-taggit==0.23.0
idna==2.8
jsonfield==2.0.2
Markdown==2.6.11
oauthlib==3.0.1
Pillow==5.3.0
Pygments==2.2.0
python3-openid==3.1.0
pytz==2018.5
requests==2.22.0
requests-oauthlib==1.2.0
urllib3==1.25.3
```

Understanding the basics of Ruby on Rails: HTTP, MVC, and Routes

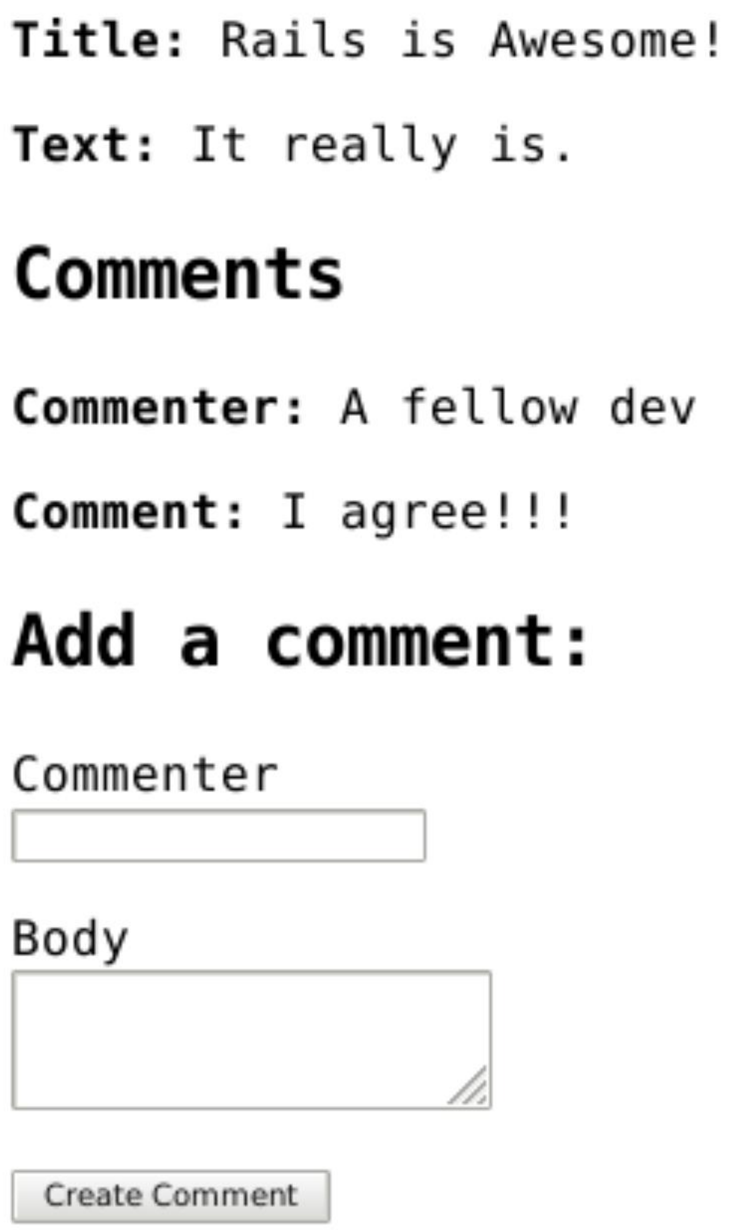


https://www.freecodecamp.org/news/understanding-the-basics-of-ruby-on-rails-http-mvc-and-routes-359b8d809c7a/

Understanding the basics of Ruby on Rails: HTTP, MVC, and Routes



Outlook of Blog Application



Title: Rails is Awesome!

Text: It really is.

Comments

Commenter: A fellow dev

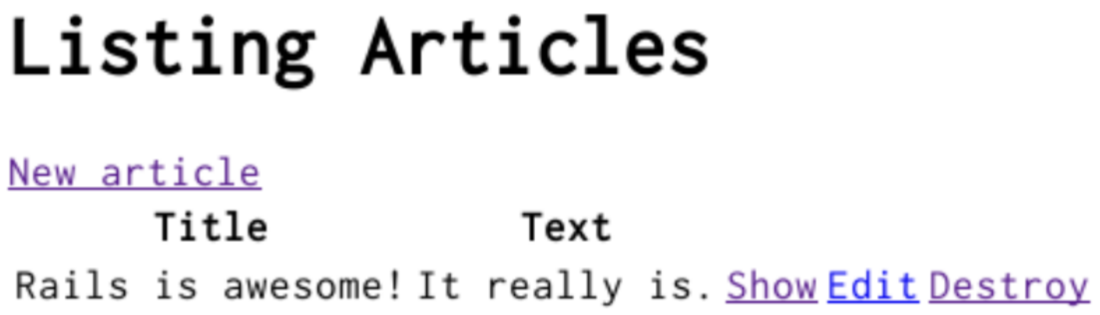
Comment: I agree!!!

Add a comment:

Commenter

Body

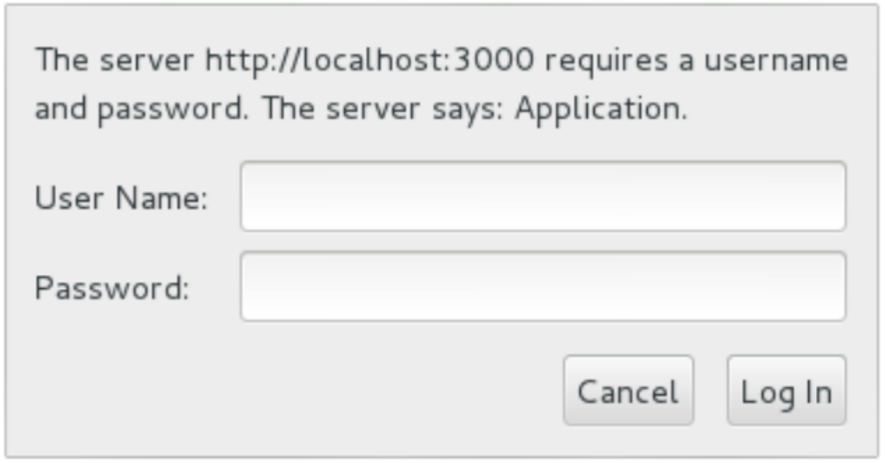
Blog Application with authentication and login



Listing Articles

New article

Title	Text			
Rails is awesome!	It really is.	Show	Edit	Destroy



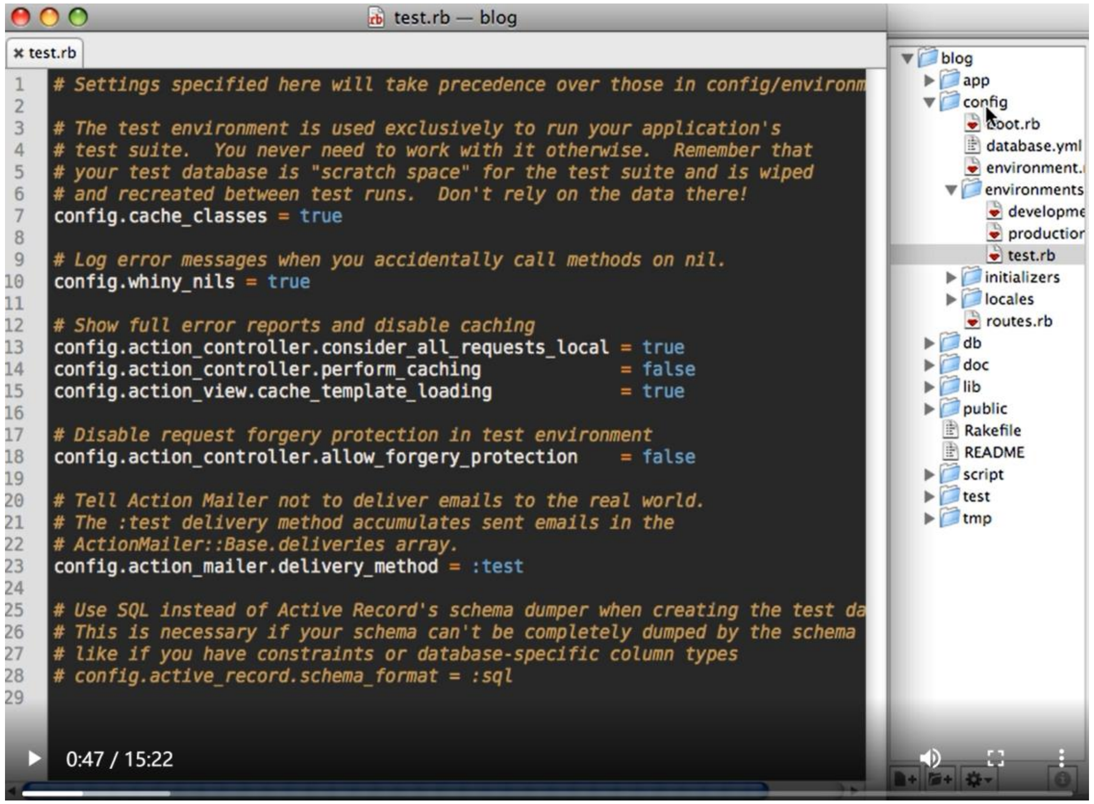
The server http://localhost:3000 requires a username and password. The server says: Application.

User Name:

Password:

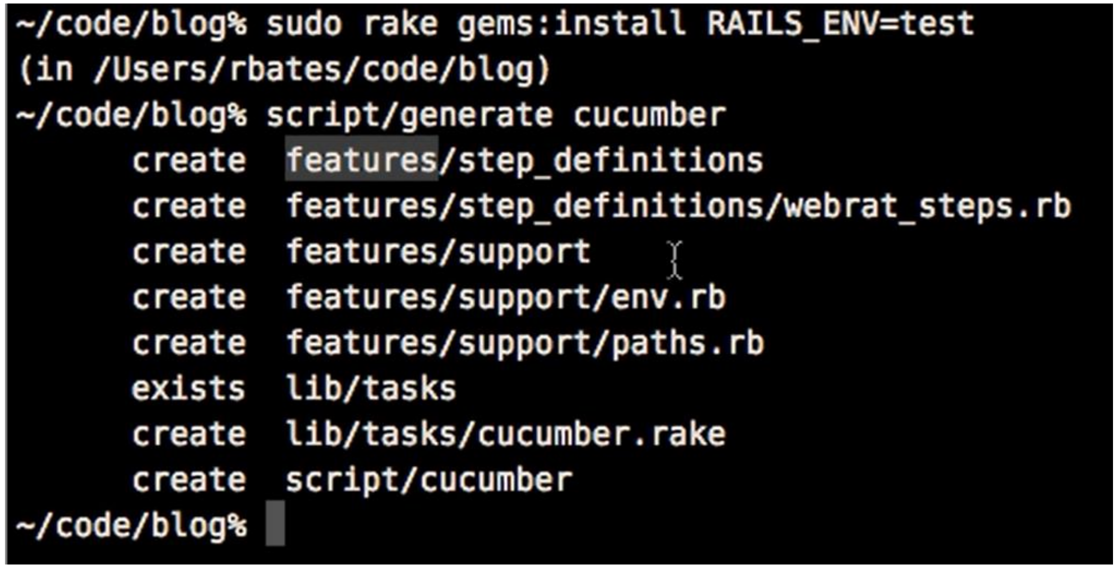
bash

```
rails blog
sudo rake gems:install RAILS_ENV=test
script/generate cucumber
cucumber features -n
script/generate rspec_model article title:string content:text
rake db:migrate
rake db:test:clone
script/generate rspec_controller articles index
```

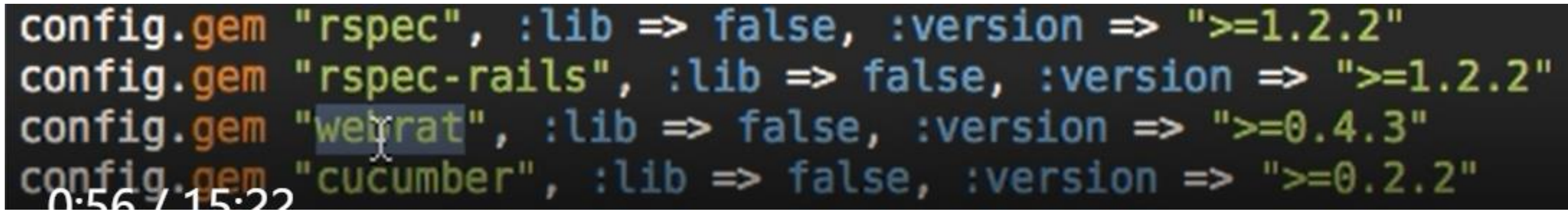


test.rb — blog

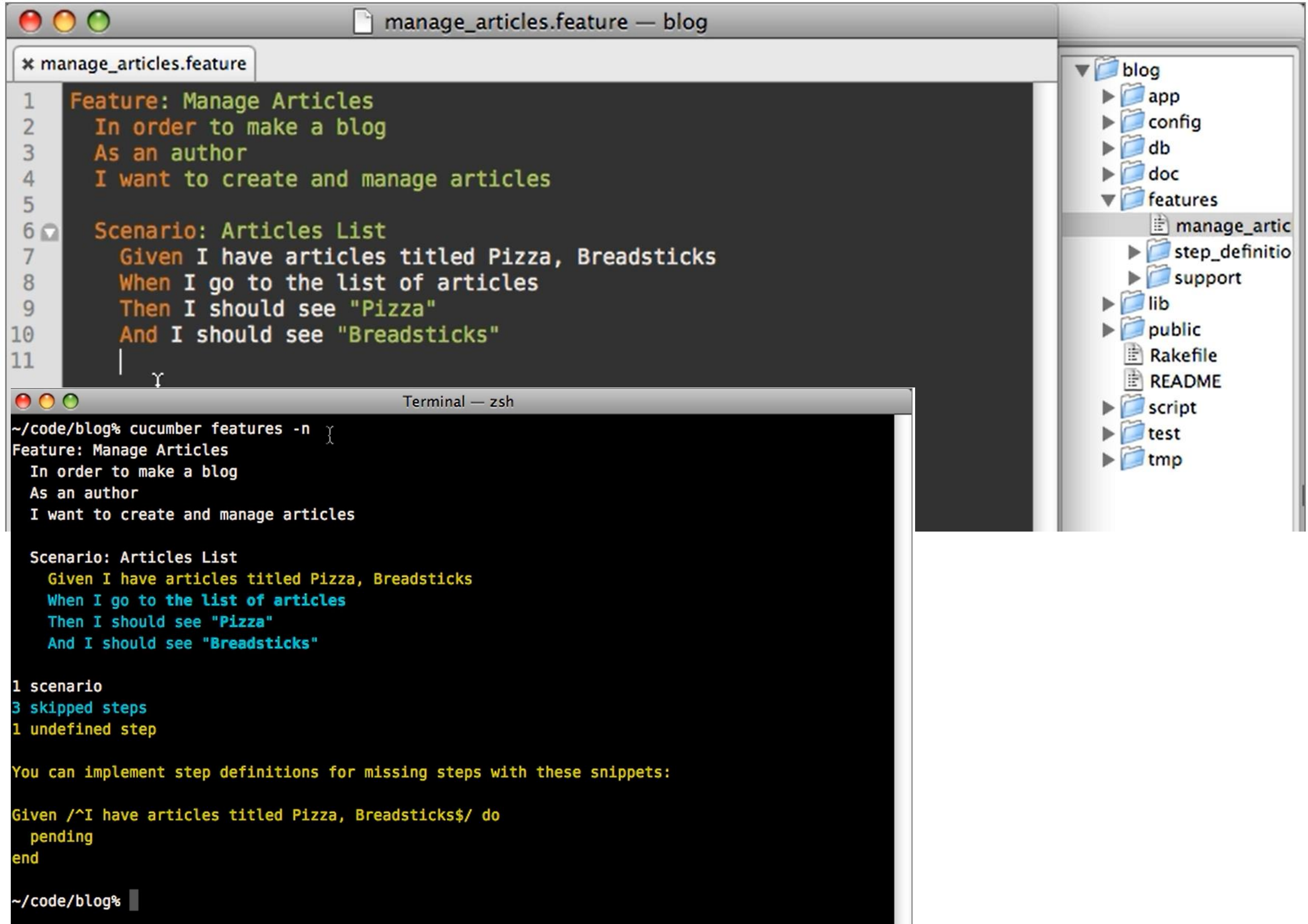
```
# Settings specified here will take precedence over those in config/environm
 
# The test environment is used exclusively to run your application's
# test suite.  You never need to work with it otherwise.  Remember that
# your test database is "scratch space" for the test suite and is wiped
# and recreated between test runs.  Don't rely on the data there!
config.cache_classes = true

# Log error messages when you accidentally call methods on nil.
config.whiny_nils = true

# Show full error reports and disable caching
config.action_controller.consider_all_requests_local = true
config.action_controller.perform_caching             = false
config.action_view.cache_template_loading            = true

# Disable request forgery protection in test environment
config.action_controller.allow_forgery_protection    = false

# Tell Action Mailer not to deliver emails to the real world.
# The :test delivery method accumulates sent emails in the
# ActionMailer::Base.deliveries array.
config.action_mailer.delivery_method = :test

# Use SQL instead of Active Record's schema dumper when creating the test da
# This is necessary if your schema can't be completely dumped by the schema
# like if you have constraints or database-specific column types
# config.active_record.schema_format = :sql
```



```
~/code/blog% sudo rake gems:install RAILS_ENV=test
(in /Users/rbates/code/blog)
~/code/blog% script/generate cucumber
      create  features/step_definitions
      create  features/step_definitions/webrat_steps.rb
      create  features/support
      create  features/support/env.rb
      create  features/support/paths.rb
      exists  lib/tasks
      create  lib/tasks/cucumber.rake
      create  script/cucumber
~/code/blog%
```



```
config.gem "rspec", :lib => false, :version => ">=1.2.2"
config.gem "rspec-rails", :lib => false, :version => ">=1.2.2"
config.gem "webrat", :lib => false, :version => ">=0.4.3"
config.gem "cucumber", :lib => false, :version => ">=0.2.2"
```

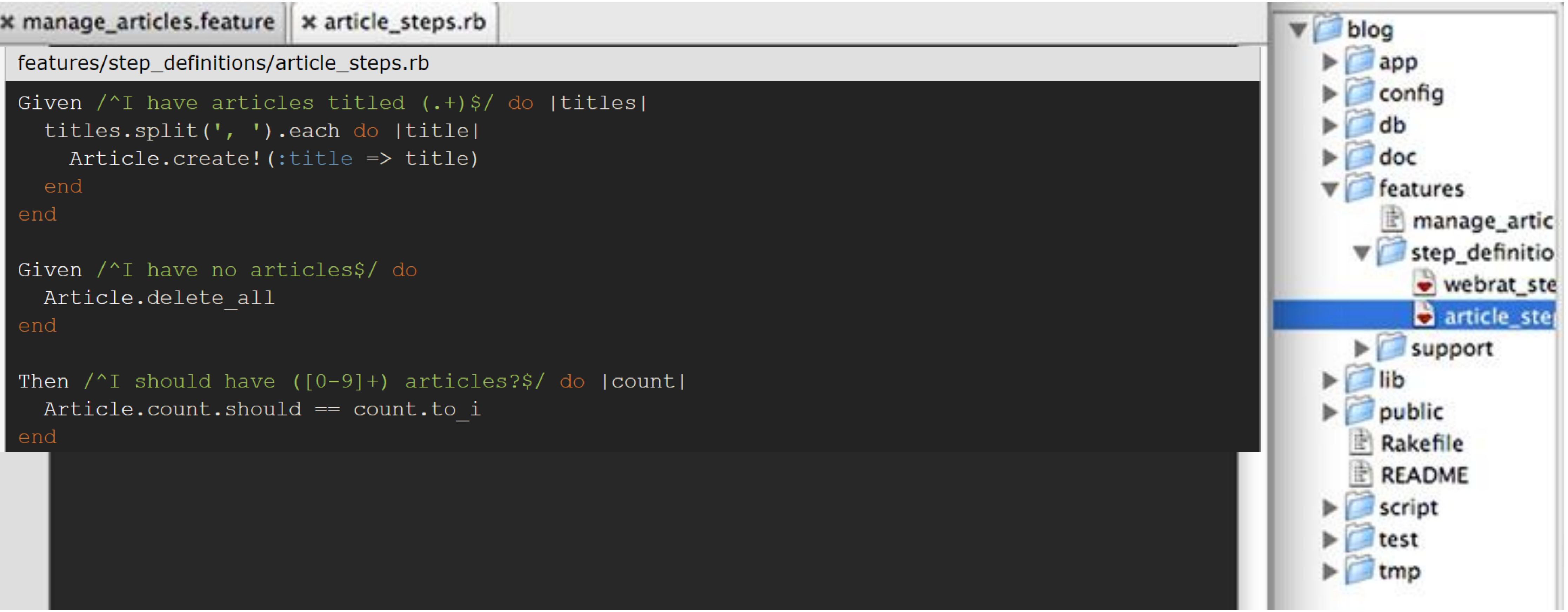


manage_articles.feature — blog

```
Feature: Manage Articles
  In order to make a blog
  As an author
  I want to create and manage articles

  Scenario: Articles List
    Given I have articles titled Pizza, Breadsticks
    When I go to the list of articles
    Then I should see "Pizza"
    And I should see "Breadsticks"
```

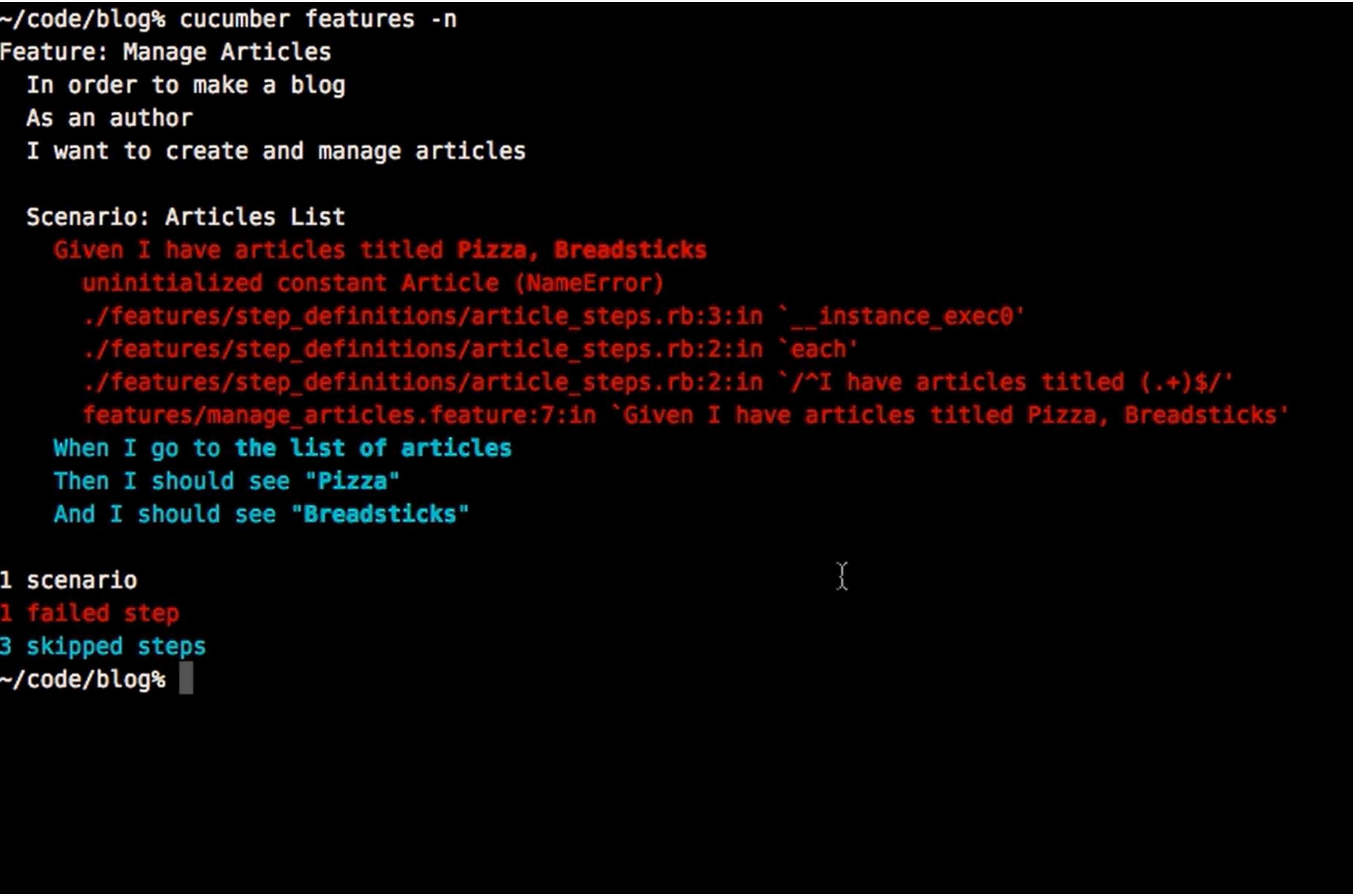
Terminal — zsh

```
~/code/blog% cucumber features -n
Feature: Manage Articles
  In order to make a blog
  As an author
  I want to create and manage articles

  Scenario: Articles List
    Given I have articles titled Pizza, Breadsticks
    When I go to the list of articles
    Then I should see "Pizza"
    And I should see "Breadsticks"

1 scenario
3 skipped steps
1 undefined step

You can implement step definitions for missing steps with these snippets:

Given /^I have articles titled Pizza, Breadsticks$/ do
  pending
end

~/code/blog%
```

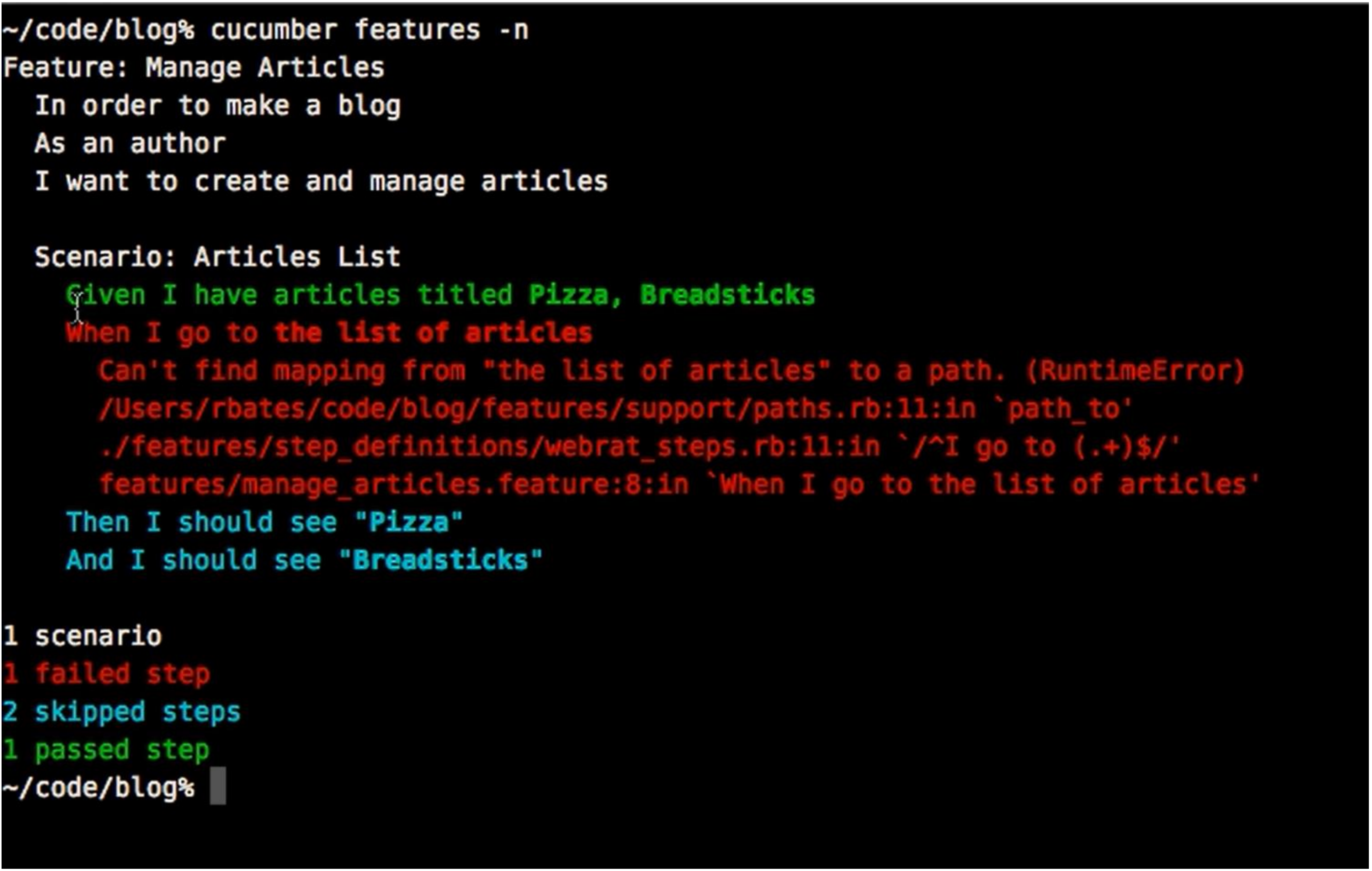


features/step_definitions/article_steps.rb

```ruby
Given /^I have articles titled (.+)$/ do |titles|
  titles.split(', ').each do |title|
    Article.create!(:title => title)
  end
end

Given /^I have no articles$/ do
  Article.delete_all
end

Then /^I should have ([0-9]+) articles?$/ do |count|
  Article.count.should == count.to_i
end
```



```
~/code/blog% cucumber features -n
Feature: Manage Articles
  In order to make a blog
  As an author
  I want to create and manage articles

  Scenario: Articles List
    Given I have articles titled Pizza, Breadsticks
      uninitialized constant Article (NameError)
      ./features/step_definitions/article_steps.rb:3:in `__instance_exec0'
      ./features/step_definitions/article_steps.rb:2:in `each'
      ./features/step_definitions/article_steps.rb:2:in `/^I have articles titled (.+)$/'
      features/manage_articles.feature:7:in `Given I have articles titled Pizza, Breadsticks'
    When I go to the list of articles
    Then I should see "Pizza"
    And I should see "Breadsticks"

1 scenario
1 failed step
3 skipped steps
~/code/blog%
```



```
~/code/blog% cucumber features -n
Feature: Manage Articles
  In order to make a blog
  As an author
  I want to create and manage articles

  Scenario: Articles List
    Given I have articles titled Pizza, Breadsticks
    When I go to the list of articles
      Can't find mapping from "the list of articles" to a path. (RuntimeError)
      /Users/rbates/code/blog/features/support/paths.rb:11:in `path_to'
      ./features/step_definitions/webrat_steps.rb:11:in `/^I go to (.+)$/'
      features/manage_articles.feature:8:in `When I go to the list of articles'
    Then I should see "Pizza"
    And I should see "Breadsticks"

1 scenario
1 failed step
2 skipped steps
1 passed step
~/code/blog%
```

cucumber

```
Feature: Manage Articles
  In order to make a blog
  As an author
  I want to create and manage articles

  Scenario: Articles List
    Given I have articles titled Pizza, Breadsticks
    When I go to the list of articles
    Then I should see "Pizza"
    And I should see "Breadsticks"

  Scenario: Create Valid Article
    Given I have no articles
    And I am on the list of articles
    When I follow "New Article"
    And I fill in "Title" with "Spuds"
    And I fill in "Content" with "Delicious potato wedges!"
    And I press "Create"
    Then I should see "New article created."
    And I should see "Spuds"
    And I should see "Delicious potato wedges!"
    And I should have 1 article
```

config/environments/test.rb

```ruby
config.gem "rspec", :lib => false, :version => ">=1.2.2"
config.gem "rspec-rails", :lib => false, :version => ">=1.2.2"
config.gem "webrat", :lib => false, :version => ">=0.4.3"
config.gem "cucumber", :lib => false, :version => ">=0.2.2"
```

features/step_definitions/article_steps.rb

```ruby
Given /^I have articles titled (.+)$/ do |titles|
  titles.split(', ').each do |title|
    Article.create!(:title => title)
  end
end

Given /^I have no articles$/ do
  Article.delete_all
end

Then /^I should have ([0-9]+) articles?$/ do |count|
  Article.count.should == count.to_i
end
```

articles_controller.rb

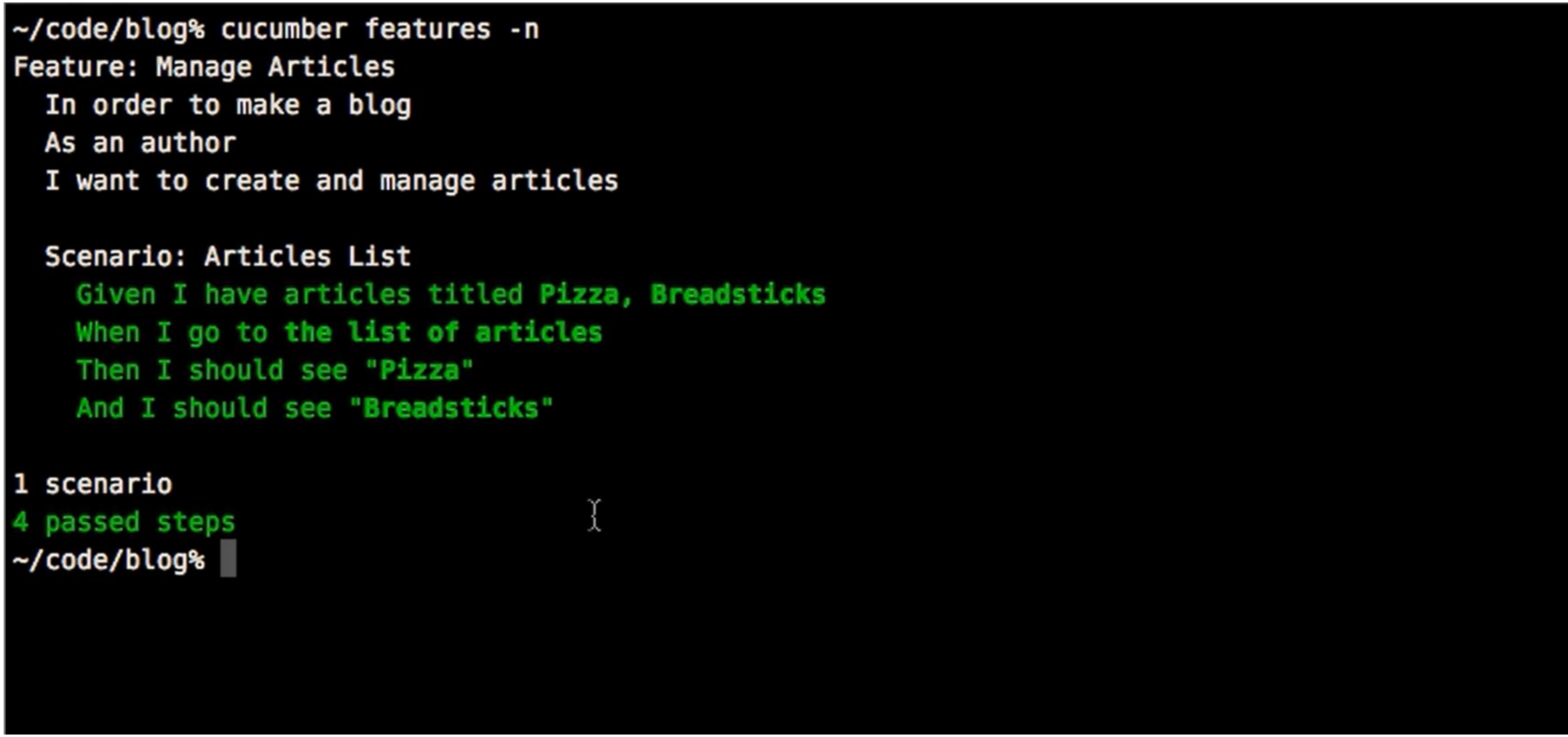
```ruby
def index
  @articles = Article.all
end

def new
  @article = Article.new
end

def create
  @article = Article.create!(params[:article])
  flash[:notice] = "New article created."
  redirect_to articles_path
end
```

features/support/paths.rb

```ruby
def path_to(page_name)
  case page_name

  when /the homepage/
    root_path
  when /the list of articles/
    articles_path

  # Add more page name => path mappings here

  else
    raise "Can't find mapping from \"#{page_name}\" to a path."
  end
end
```

index.html.erb

```erb
<%= flash[:notice] %>
<% for article in @articles %>
  <p><%=h article.title %></p>
  <p><%=h article.content %></p>
<% end %>
<p><%= link_to "New Article", new_article_path %></p>
```



```
~/code/blog% cucumber features -n
Feature: Manage Articles
  In order to make a blog
  As an author
  I want to create and manage articles

  Scenario: Articles List
    Given I have articles titled Pizza, Breadsticks
    When I go to the list of articles
    Then I should see "Pizza"
    And I should see "Breadsticks"

1 scenario
4 passed steps
~/code/blog%
```

总结

- 基于BDD软件开发是“先代码后测试、自动化测试、TDD”的演化过程。
- BDD的软件开发除了有工具支撑，还要改变团队工作模式。
- 软件测试框架和应用软件框架是BDD的技术基础。