Object Oriented WEB Programming Using Ruby

Day 5: 17/May/2012

Validation, Error Messages

Today's Goal

- Implement the Validation method
  - to reject improper input

Purpose of Validation

- Validations are used to ensure that only valid data is saved into our database.
- Why Validate?
  - At the Entrance of data, reject invalid data
    - Example: Age 500?
    - Stock sales order: 500,000 shares at 1 yen!?
    - Pizza order : 17 pizzas of 1 inch!?
  - To avoid logical Error and troubles, “unacceptable values” are programmed to be blocked.

http://guides.rubyonrails.org/active_record_validations_callbacks.html#why-use-validations

Validation and Verification

- Validation is to prove data in the context
  - Logical check

- Verification is the check of format and
  - symbolic check
  - CD and DVD’s read after write check
  - Physical format check
  - Protocol header check sum, etc.

Design Guest Table to the System

Our goal is to develop the Problem Solving Engine.

It is desirable to let any ‘guest’ write into the causes and solutions’ links to certain problem, to collect wisdom.

But, the solutions may vary depending to ages, sexes, occupations, and such.

So first, we give the Guest Table the field of age and sex, for the beginning.

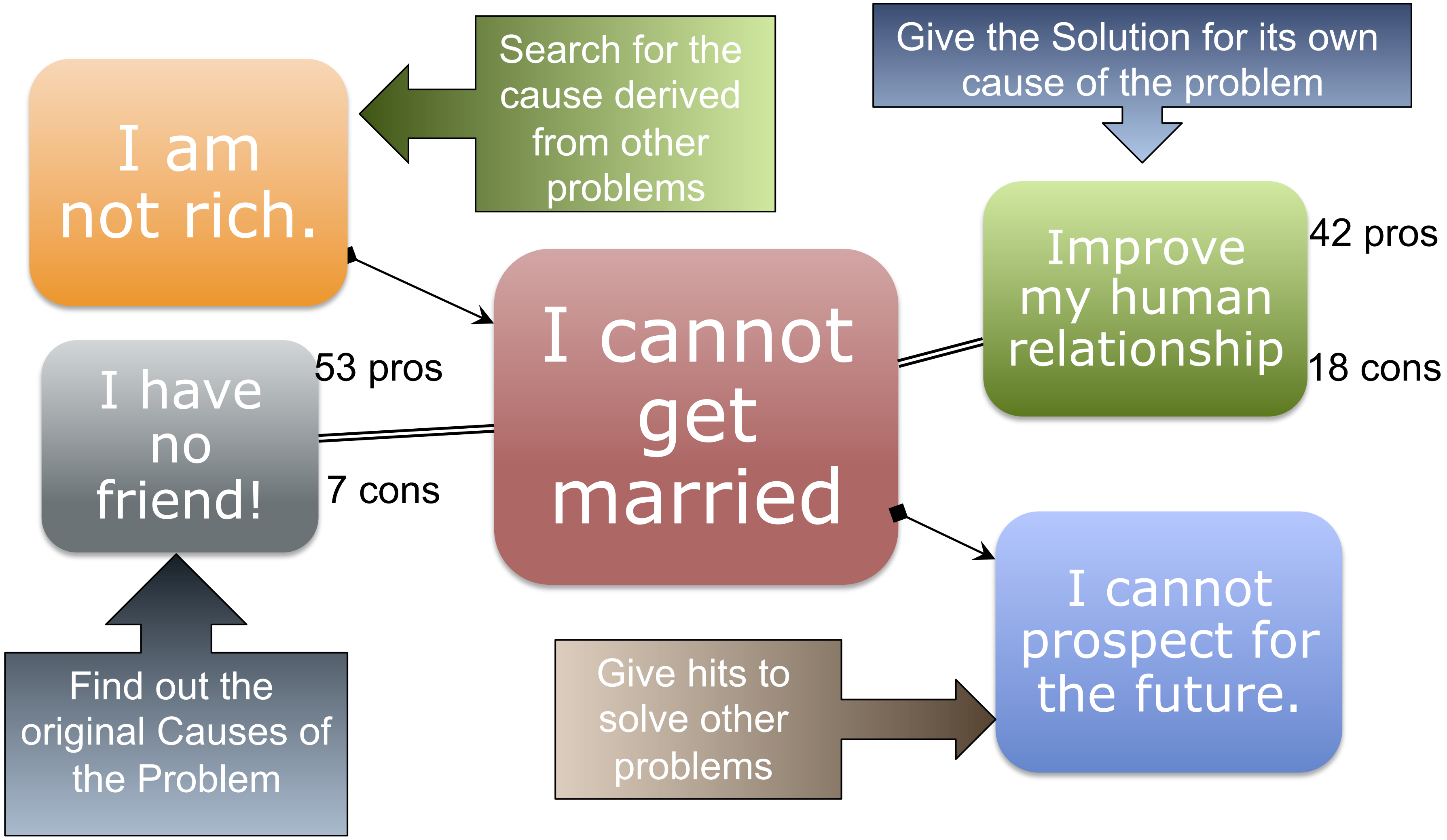
(Is there any expanded version coming?)

The Pros and Cons

We also give Causes and Solutions field the counter of the pros and cons.

Let many guests to click the buttons for pro and con, (desirably only once per person.)

Example of pros and cons


Search for the cause derived from other problems
Give the Solution for its own cause of the problem
I am not rich.
I cannot get married
Improve my human relationship
42 pros
18 cons
53 pros
I have no friend!
7 cons
I cannot prospect for the future.
Give hits to solve other problems
Find out the original Causes of the Problem

Example of pros and cons (2)

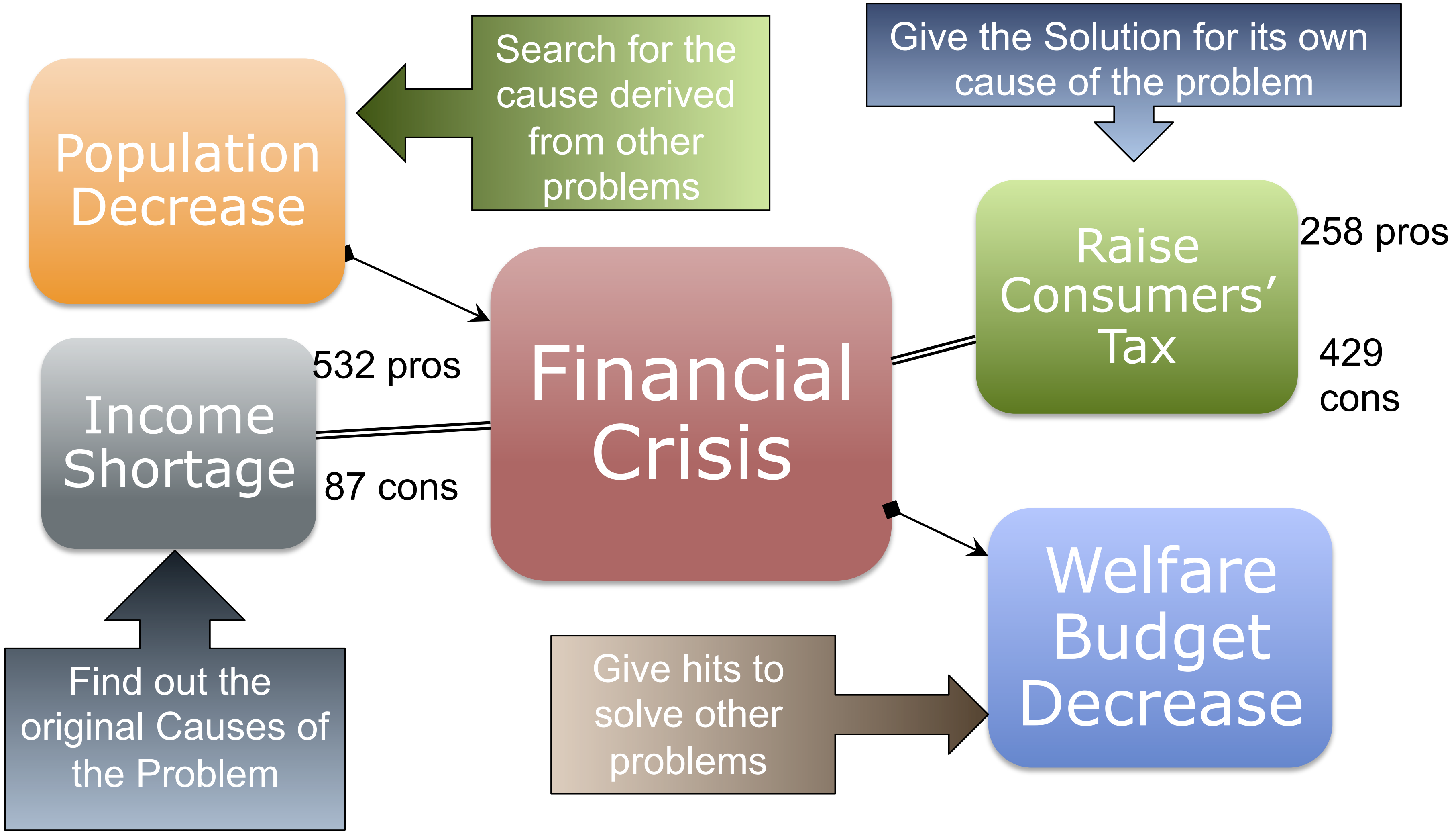

Search for the cause derived from other problems
Give the Solution for its own cause of the problem
Population Decrease
Raise Consumers' Tax
258 pros
429 cons
Financial Crisis
532 pros
Income Shortage
87 cons
Welfare Budget Decrease
Find out the original Causes of the Problem
Give hits to solve other problems

Table Design for [Guest]

Guest

- can be anonymous, but should have "login id", to collect pros and cons fairly.

Let "Login ID" be "mail address," for self registration.

Fields:

- Age (Integer), and Sex (Integer)

Table Design for [Problem]

Problem

has a title field (string,)

a content field (text,)

and the proposer’s guest ID (link.)

Table Design for [Causes]

Essential Cause should be “Facts.”

Cause

- should have a field of ‘fact’ (text,)
- a counter for pro (integer,)
- a counter for con (integer,)
- and a link to the solution (link.)

Table Design for [Solution]

Solution is an “Action.”

Solution

- should have a field of ‘action’ (text,)
- a counter for pro (integer,)
- a counter for con (integer,)
- and a link to the solution (link.)

Table Design for [vote]

‘Vote’ is the special feature of this system, to collect wisdom of visitors.

Vote table is the record of guests’ participation.

Vote

should have a field of guest ID (link,)

a flag of pro or con (integer,)

a link to which ‘cause’ or ‘solution’ vote (link.)

Self link of [Problems]

One ‘problem’ could be a cause of another problem, or it could lead to another problem.

So problem table should have ‘to link’ and ‘from link’ between records.

Today's Goal

Install Guest Table which has fields of

login (email address) :string,

age :integer,

sex :integer,

With the Validation of input values.

Password field should be encrypted, but we will use a gem for login authentication later. Until then, we do not use password.

Validation Rule for guest

login:string

should be a format of email address, which contain only one ‘@’, and the other letters should A-Z, a-z, ‘.’, _, %, +, or ‘-’.

(But we can use mail gem to validate email address.)

age:integer

valid when it is between 1 and 130!

sex: integer

valid when it is 1 or 2, but let it input with ‘radio button.’

Scaffolding

The scaffolding command is

```
rails g scaffold guest login:string
age:integer sex:integer
```

Please note that the command should be typed in one line.

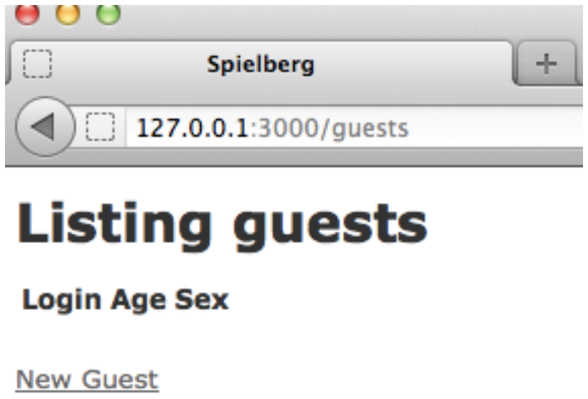
Migration and Test Run

Let’s migrate the database, and test run

```
rake db:migrate
```

and test run

```
rails server
```



Listing guests

Login Age Sex

New Guest

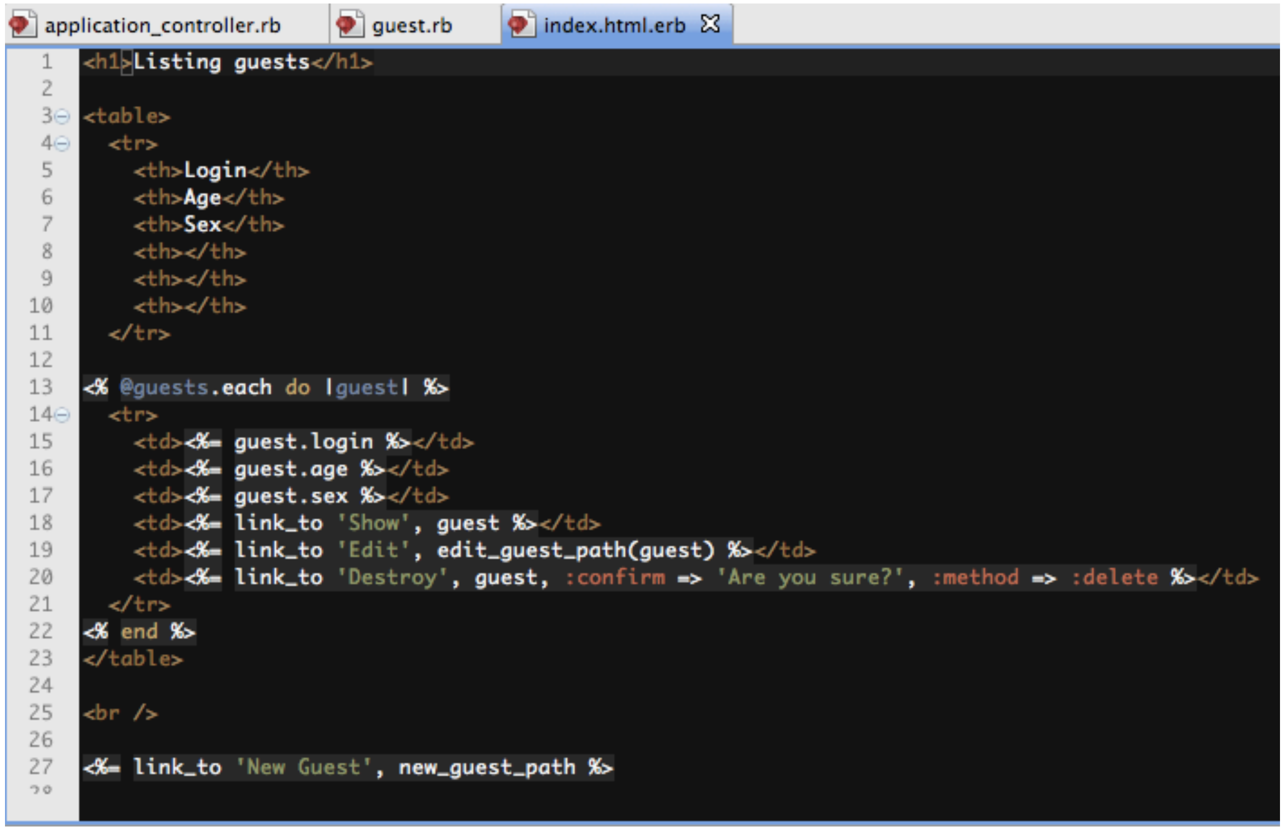
Open the WEB page with the following URL

http://127.0.0.1:3000/guests

app/views/guests/index.html.erb

Now we obtain new 'Mini Application' to register guests.

The Front WEB page is given in app/views/guests/index.html.erb.



```erb
<h1>Listing guests</h1>

<table>
  <tr>
    <th>Login</th>
    <th>Age</th>
    <th>Sex</th>
    <th></th>
    <th></th>
    <th></th>
  </tr>

<% @guests.each do |guest| %>
  <tr>
    <td><%= guest.login %></td>
    <td><%= guest.age %></td>
    <td><%= guest.sex %></td>
    <td><%= link_to 'Show', guest %></td>
    <td><%= link_to 'Edit', edit_guest_path(guest) %></td>
    <td><%= link_to 'Destroy', guest, :confirm => 'Are you sure?', :method => :delete %></td>
  </tr>
<% end %>
</table>

<br />

<%= link_to 'New Guest', new_guest_path %>
```

Where are database tables?

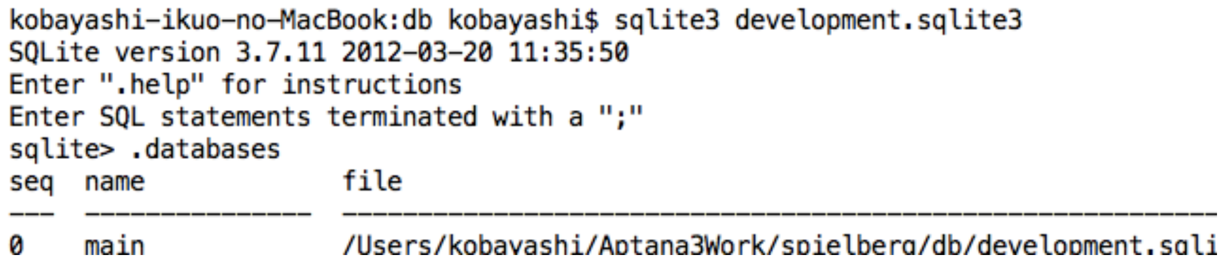
Sqlite3 database has been create in (project directory) \db

Change directory to (project) \ db

See if there is a file ‘development.sqlite3.’

Then, type command

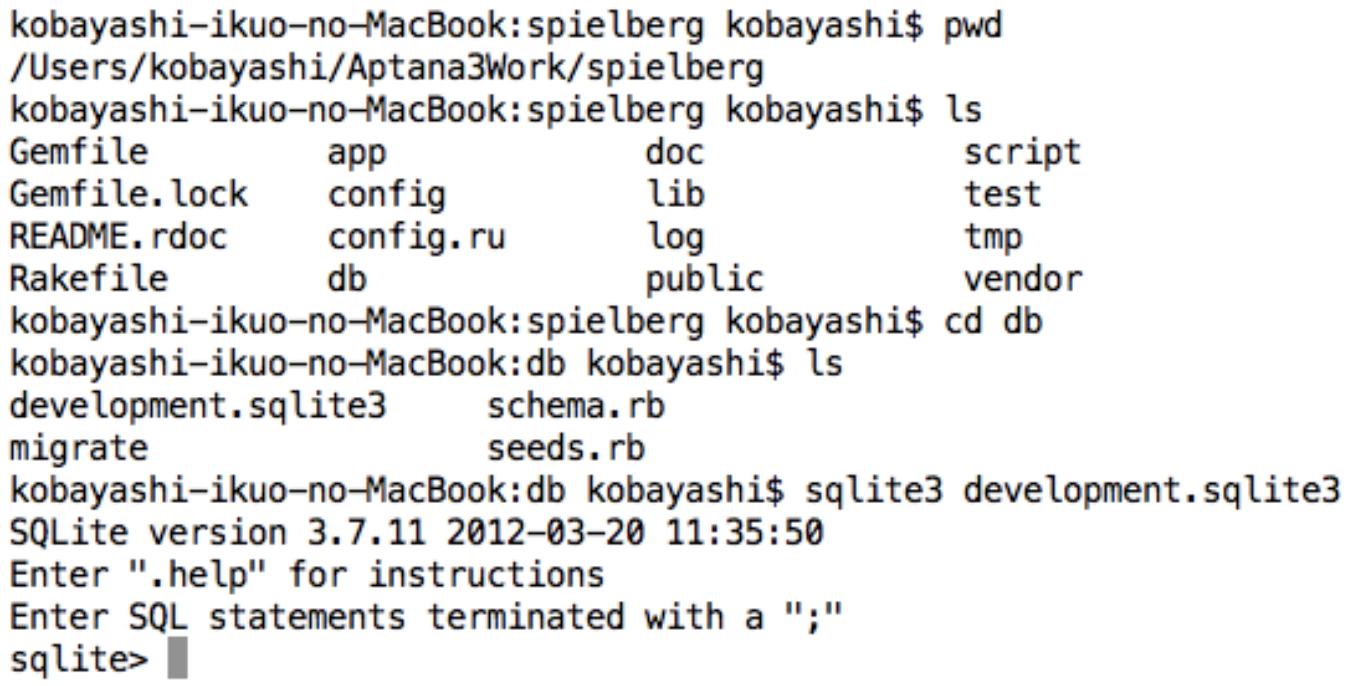
sqlite3 development.sqlite3



```
kobayashi-ikuo-no-MacBook:db kobayashi$ sqlite3 development.sqlite3
SQLite version 3.7.11 2012-03-20 11:35:50
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .databases
seq  name             file
---  ---------------  ----------------------------------------------------------
0    main             /Users/kobayashi/Aptana3Work/spielberg/db/development.sqli
```

Sqlite3 command

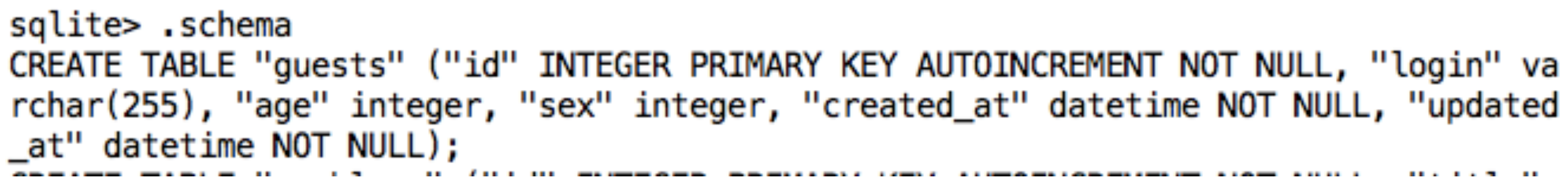
- You will see the message that SQL statements should be followed by ‵;.’
- Enter ‵.help’ for Instruction.
- Enter ‵.databases’ to get main database.
- Select * from guests; to see the contents of guests table.



```
kobayashi-ikuo-no-MacBook:spielberg kobayashi$ pwd
/Users/kobayashi/Aptana3Work/spielberg
kobayashi-ikuo-no-MacBook:spielberg kobayashi$ ls
Gemfile         app             doc             script
Gemfile.lock    config          lib             test
README.rdoc     config.ru       log             tmp
Rakefile        db              public          vendor
kobayashi-ikuo-no-MacBook:spielberg kobayashi$ cd db
kobayashi-ikuo-no-MacBook:db kobayashi$ ls
development.sqlite3     schema.rb
migrate                 seeds.rb
kobayashi-ikuo-no-MacBook:db kobayashi$ sqlite3 development.sqlite3
SQLite version 3.7.11 2012-03-20 11:35:50
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> 
```

Guests Schema in the Database

- Enter `.schema guests' to see schema.
- We did not specify ID of records during the scaffolding and migration, but an "id" field is generated.
- It is especially important for "LINK."
  - If we write "guest_id"(Singular_id,) in other tables, the field will mean the link to the Guests table, to look up a record with the given "ID" value.



```
sqlite> .schema
CREATE TABLE "guests" ("id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "login" va
rchar(255), "age" integer, "sex" integer, "created_at" datetime NOT NULL, "updated
_at" datetime NOT NULL);
```

Validators

For detail, connect rails documentation site:

http://api.rubyonrails.org/

and search for

"ActiveModel::Validations::HelperMethods"

Language Style of Ruby

Open app/models/guest.rb file.

```
class class_name < Inherited_class
End
```

is ruby description of Class definition.

```
class Guest < ActiveRecord::Base
  attr_accessible :age, :login, :sex
end
```

tells that the table Guest inherits ActiveRecord::Base.

This is a framework which allows users to modify.

Add Validation Method

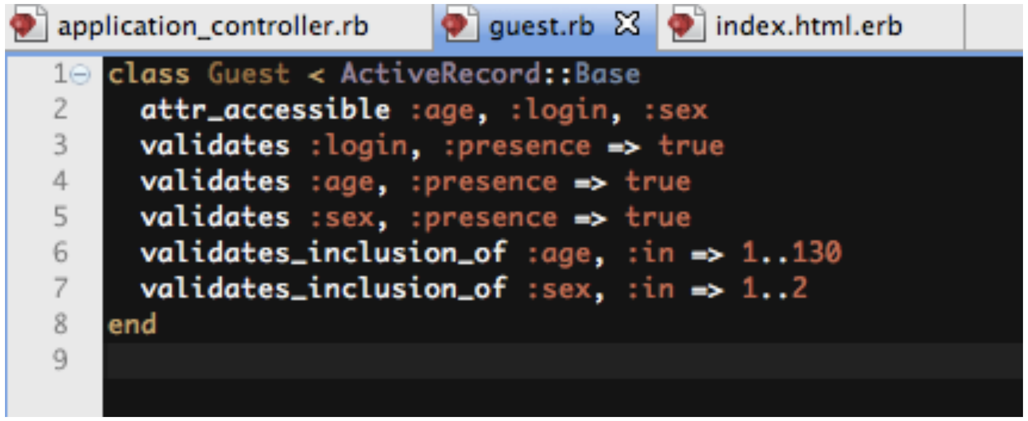
- Open app/models/guest.rb file.
- Then, add the following lines;

```
validates :login, :presence => true
validates :age, :presence => true
validates :sex, :presence => true
validates_inclusion_of :age, :in => 1..130
validates_inclusion_of :sex, :in => 1..2
```

- Validators are added to Active Record
  - ActiveRecord has an important roll to bridge between database and controllers.

app/models/guest.rb

It should look like:



```ruby
class Guest < ActiveRecord::Base
  attr_accessible :age, :login, :sex
  validates :login, :presence => true
  validates :age, :presence => true
  validates :sex, :presence => true
  validates_inclusion_of :age, :in => 1..130
  validates_inclusion_of :sex, :in => 1..2
end
```

Validate :login, :presence => true

tells that :login field require value.

Validate_inclusion_of :age, :in => 1..130

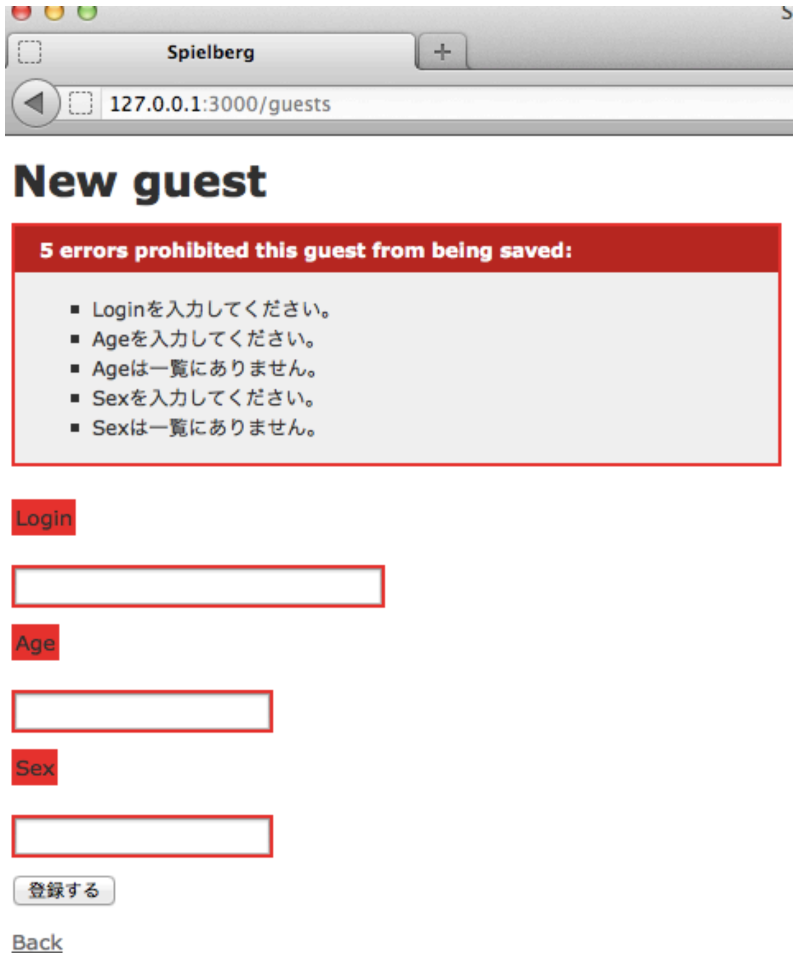
tell that the value of :age should be within the range from 1 to 130.

Test with validators

Run server again with validators.

Error Message will appear.

Validators check the existence of value, range, and such.



New guest

5 errors prohibited this guest from being saved:

- Loginを入力してください。
- Ageを入力してください。
- Ageは一覧にありません。
- Sexを入力してください。
- Sexは一覧にありません。

Set the Multi-lingual Support

Review of the lesson of last week;

Open app/views/guests/index.html.erb file.

Modify String literals into ruby translation.

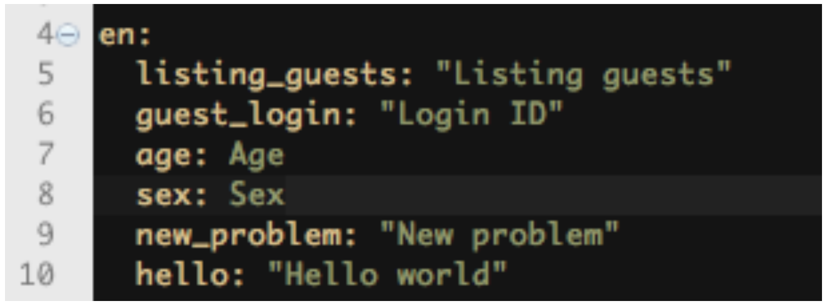
Ex. ‘Listing guests’ to

```
<%= t :listing_guests %>
```

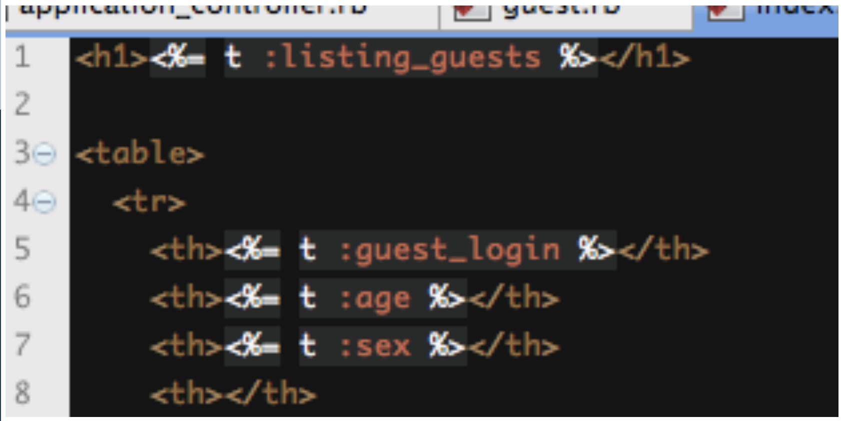
Then prepare :listing_guests symbol in both ja.yml and en.yml file.

Modify titles of index page

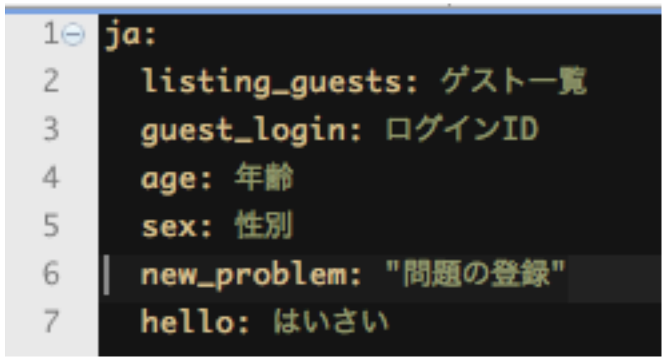
Modify
  app/views/guests/index.html.erb
  config/locales/ja.yml
  config/locales/en.yml



```
en:
  listing_guests: "Listing guests"
  guest_login: "Login ID"
  age: Age
  sex: Sex
  new_problem: "New problem"
  hello: "Hello world"
```



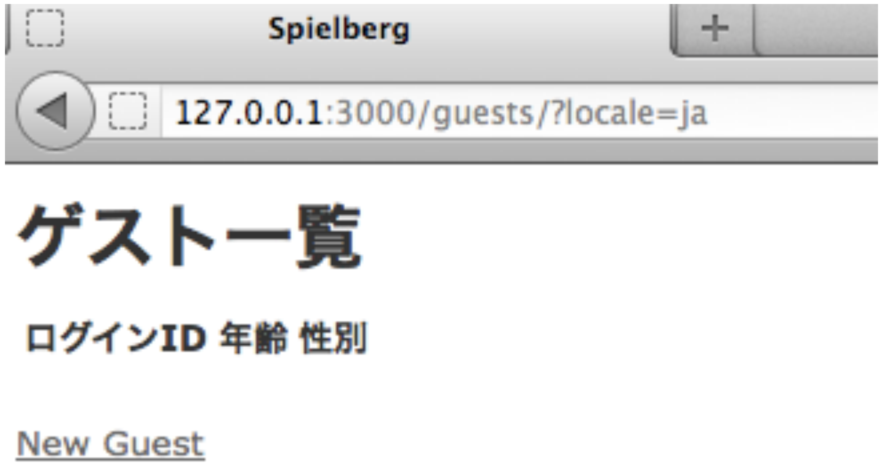
```
<h1><%= t :listing_guests %></h1>

<table>
  <tr>
    <th><%= t :guest_login %></th>
    <th><%= t :age %></th>
    <th><%= t :sex %></th>
    <th></th>
```



```
ja:
  listing_guests: ゲスト一覧
  guest_login: ログインID
  age: 年齢
  sex: 性別
  new_problem: "問題の登録"
  hello: はいさい
```

Check if you switch locales

By adding /?locale=en or /?locale=ja,
you can switch language environment.



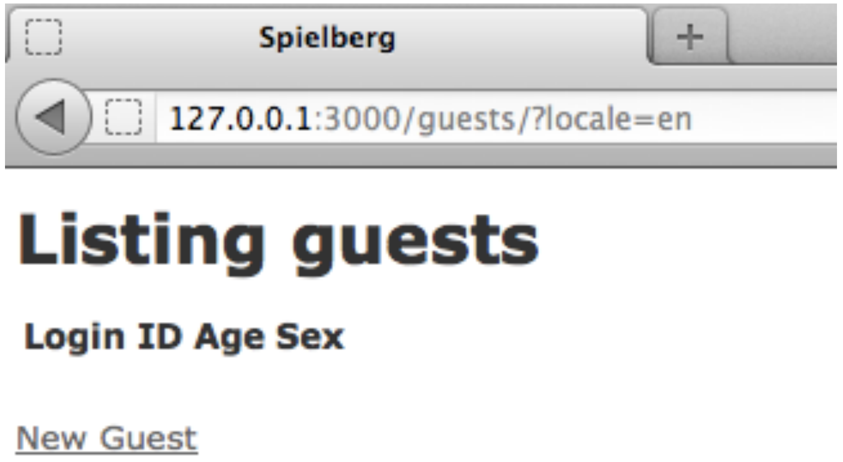
Spielberg

127.0.0.1:3000/guests/?locale=ja

ゲスト一覧

ログインID 年齢 性別

New Guest



Spielberg

127.0.0.1:3000/guests/?locale=en

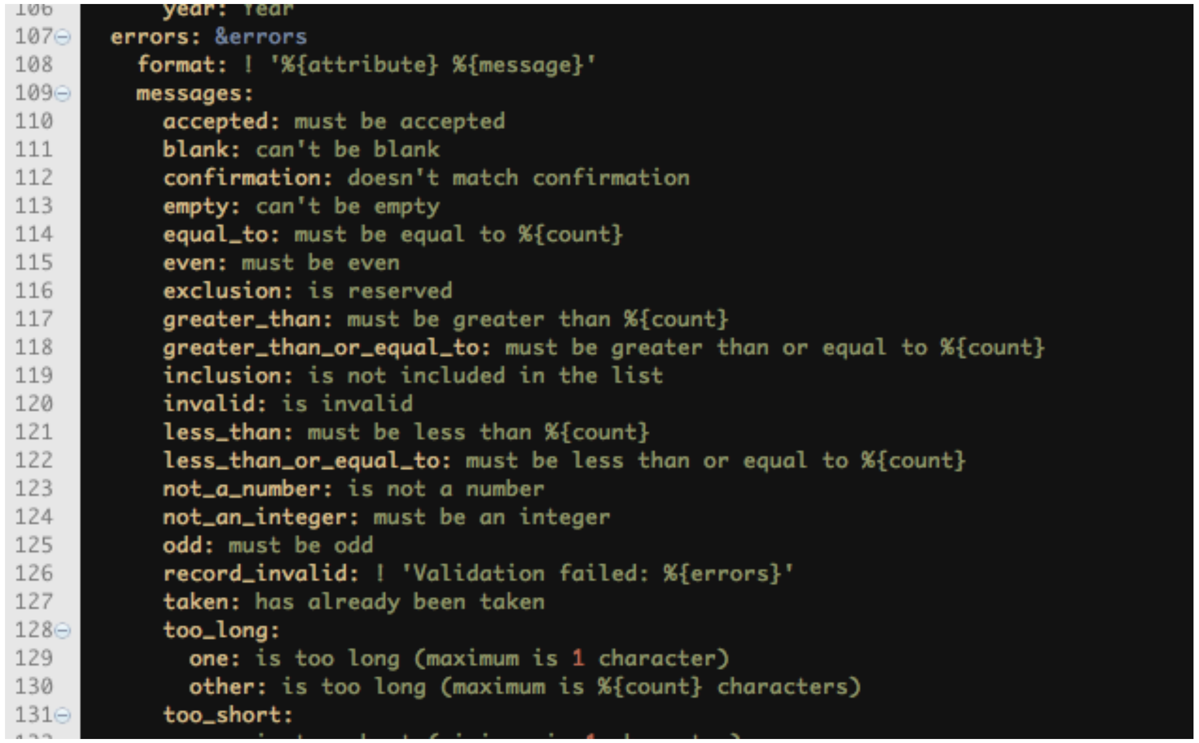
Listing guests

Login ID Age Sex

New Guest

Error messages

- Please scroll to below and see en.yml or ja.yml. You will see
  - errors:messages:empty and such.



```
      year: Year
  errors: &errors
    format: ! '%{attribute} %{message}'
    messages:
      accepted: must be accepted
      blank: can't be blank
      confirmation: doesn't match confirmation
      empty: can't be empty
      equal_to: must be equal to %{count}
      even: must be even
      exclusion: is reserved
      greater_than: must be greater than %{count}
      greater_than_or_equal_to: must be greater than or equal to %{count}
      inclusion: is not included in the list
      invalid: is invalid
      less_than: must be less than %{count}
      less_than_or_equal_to: must be less than or equal to %{count}
      not_a_number: is not a number
      not_an_integer: must be an integer
      odd: must be odd
      record_invalid: ! 'Validation failed: %{errors}'
      taken: has already been taken
      too_long:
        one: is too long (maximum is 1 character)
        other: is too long (maximum is %{count} characters)
      too_short:
```

Report themes for today

none

Prepare for the Next Week

We will learn Test Driven Development.