Homework 4: MVC Model

For this homework, you will create simple model classes in Rails, write tests to run them, and practice using the basic version control features of Git.

You will do this homework as a team; however, each member of your team will be responsible for the completion of a particular task.

Each team member will choose one task from the list below to complete. All team members must do a different task. If your team has only three members, then ignore Task 4.

IMPORTANT: Be sure to add your university username (e.g., mine is "sdflming") in a comment at the top of your model class file. (Otherwise, how will we know which work is yours?)

How to submit your team's work

Before you can submit, all team members must have merged their code into the master branch and pushed the updates to GitHub. If a team member does not complete his/her work on time, you may submit without his/her contribution.

To submit your team's work, you must "tag" the current commit in the master branch:

```
$ git tag -a hw4v1 -m 'Tagged Homework 4 submission (version 1)'
$ git push origin --tags
```

To grade your work, I will check out the appropriate tag, and run it on my machine.

Note that if for some reason you need to update your submission, simply repeat the tagging process, but increment the version number (e.g., hw4v2, hw4v3, hw4v4, etc.).

The Tasks

The tasks are given on the following pages. For each of the following tasks:

- Create model classes as per the given UML class diagram.
  - Use `rails g model ...`
- Add attribute validations to one of the classes as specified. Here is a page to help you:
  - http://guides.rubyonrails.org/v5.0.1/active_record_validations.html
- Write unit tests as follows for both classes. For each attribute write one test for a valid value of the attribute. For each validation of the attribute write one test that violates the validation rule. For example, if a string attribute has a length validation and a presence validation, you will have 3 tests for that attribute: one with a valid string, one with a string of invalid length, and one with an empty string. In the case of an "allow_blank" validation, include a test for a blank value. Here are couple pages to help you:
  - http://guides.rubyonrails.org/v5.0.1/active_record_basics.html#crud-reading-and-writing-data
  - http://guides.rubyonrails.org/v5.0.1/testing.html

Task 1

Model classes to create:

Planet
name : string color : string nasa_data : string diameter : integer

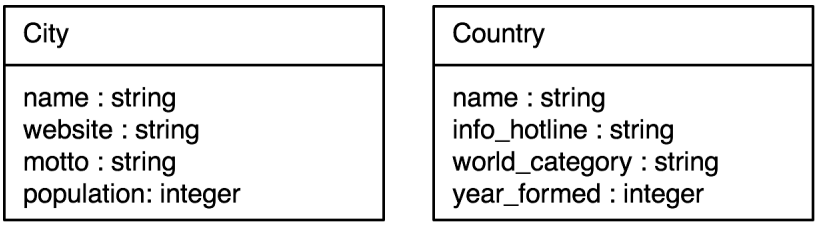
Continent
name : string area : integer population : integer description : text

Add these validations to class **Planet**:

- `length`
  - **name** must be 30 characters or less.
- `format`
  - **nasa_data** must follow the general format of a basic HTTP or HTTPS URL.
- `inclusion`
  - color must be “red”, “orange”, “yellow”, “green”, “blue”, “indigo”, or “violet”.
- `uniqueness`
  - **name** must be unique.
- `presence`
  - **name**, **color**, and **diameter** must have values.
- `allow_blank`
  - **nasa_data** may be blank.
- `numericality`
  - **diameter** must be greater than 0.

Task 2

Model classes to create:

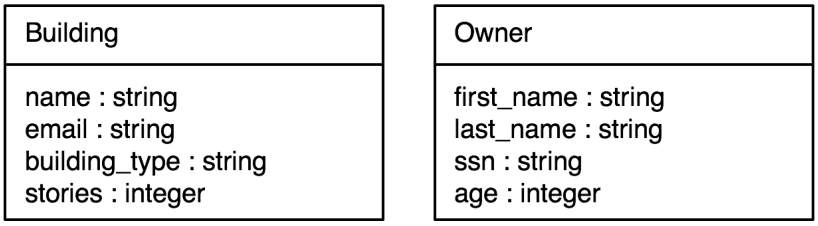


Add these validations to class **Country**:

- `length`
  - **name** must be 50 characters or less.
- `format`
  - **info_hotline** must follow the general format of a 10-digit telephone number (e.g.,
  - "123-456-7890").
- `inclusion`
  - **world_category** must be "first", "second", or "third".
- `uniqueness`
  - **name** must be unique.
- `presence`
  - **name** and **world_category** must have values.
- `allow_blank`
  - **info_hotline** and **year_formed** may be blank.
- `numericality`
  - **year_formed** must be less than or equal to the current year (whatever year that might be).

Task 3

Model classes to create:



Add these validations to class **Building**:

- `length`
  - **name** must be 100 characters or less.
- `format`
  - **email** must follow the general format of an email address.
- `inclusion`
  - **building_type** must be “agricultural”, “commercial”, “residential”, “medical”, “educational”, “government”, “industrial”, “military”, “storage”, “religious”, “transport”, “infrastructure”, “power”, or “other”.
- `uniqueness`
  - **name** must be unique.
- `presence`
  - **name**, **building_type**, and **stories** must have values.
- `allow_blank`
  - **email** may be blank.
- `numericality`
  - **stories** must be greater than 0 and less than 1000 (because no building could possibly that tall, could it?).

Task 4

Model classes to create:

Statue
name : string date_created : string medium : string weight : integer

Park
name : string park_type : string year_created : integer area : integer

Add these validations to class **Statue**:

- `length`
  - **name** must be no more than 75 characters.
- `format`
  - **date_created** must follow this date format: YYYY-MM-DD (e.g., “2017-02-25”).
- `inclusion`
  - **medium** must be “alabaster”, “bronze”, “clay”, “hardstone”, “ivory”, “marble”, “terracotta”, “steel”, or “other”.
- `uniqueness`
  - **name** must be unique.
- `presence`
  - **name** and **medium** must have values.
- `allow_blank`
  - **date_created** and **weight** may be blank.
- `numericality`
  - **weight** must be greater than 0 and less than 2,000,000 (because what statue could possibly be that heavy, right?).