Homework 5: MVC Controller

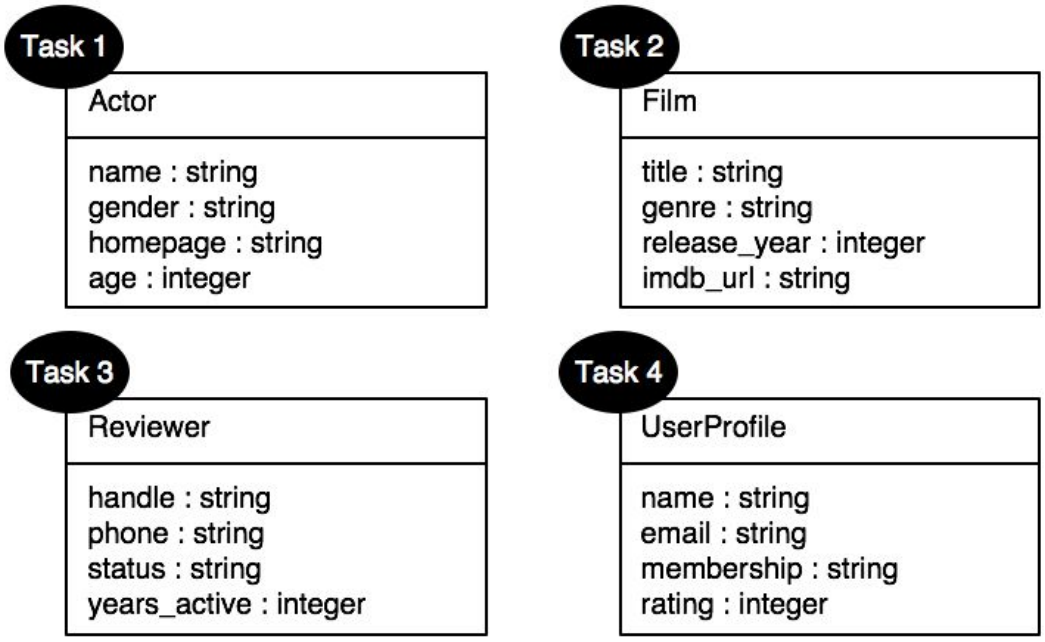
For this homework, you will build upon the scaffolded model classes that you created in Homework 4 by adding custom controller actions and views. Moreover, you will continue to practice with the version-control system, Git.

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.

The Tasks

For this homework, there will be four tasks (Tasks 1 through 4), each of which will build upon the corresponding task from Homework 4. Thus, Task 1 will work with the actor-related code, Task 2 will work with the film-related code, etc. As a reminder, here are the model classes and their associated tasks from Homework 4:



Overall, the tasks for this homework are all very similar to one another, with their small differences being related to their different model classes. In cases where the specifics of the model class don't matter, we will use the model class name *WildPig* below. Of course, you should replace all references to WildPig with the name of the model class associated with your task.

Each task must follow these requirements:

- ❏ **Leave original scaffolds unchanged**: The features that you created last homework (e.g., *index*, *create*, *update*, etc. pages) should not be changed in any way during this homework. Your additions for this homework must not affect them.

- ❏ **Add seed data**: Using the *db/seeds.rb* file, seed your database with five valid example objects of the model class associated with your task. Use example names and values that sound realistic.

- ❏ **Add an *inventory* action and view**: Add to the controller associated with your task's model class an action called *inventory*. Also, add an associated view with the same name (i.e., *inventory.html.erb*). This action and view should follow these requirements:

    - ❏ **Route to *inventory* action**: The route to the *inventory* action should be as follows (assuming that your model class is called *WildPig*).
        - ❏ HTTP Method: GET
        - ❏ URI Pattern: /wild_pigs/inventory
        - ❏ Prefix: wild_pig_inventory

    - ❏ **Like *index*, but different**: Your inventory action and view should be like the *index* action and view, but with the following differences:

        - ❏ **Sorted differently**: The rows of your table should be sorted as follows, depending on your model class:
            - ❏ **Actor**: ascending by name
            - ❏ **Film**: descending by imdb_url
            - ❏ **Reviewer**: ascending by handle
            - ❏ **UserProfile**: descending by email

        - ❏ **No "Show" links**: The "Show" links should be removed from the table. (Be careful to adjust the columns in the headers row.)

        - ❏ **Rename a couple other links**: The "Edit" links should be renamed "Update", and the "Destroy" links should be renamed "Delete". These links should all still function the same as before, though.

        - ❏ **New id column**: The table should have a new first column. The column should be labeled *id*, and it should contain the id value of each object displayed. Each id value should also be a hyperlink to the associated *show* page for that record.

        - ❏ **Removed columns**: The following columns should be removed from the table, depending on the model class:
            - ❏ **Actor**: gender, age
            - ❏ **Film**: genre, release_year
            - ❏ **Reviewer**: status, years_active
            - ❏ **UserProfile**: membership, rating

- ❏ **No "New Wild Pig" link**: The "New Wild Pig" link below the table should be removed (assuming the model is called *WildPig*).
- ❏ **Add create form**: A form to create a new model object must be displayed below the table. However, the form must be different from the standard *new* form in two ways:
  - ❏ **Reorder the input fields**. The input fields for this form must be in an entirely different order than the input fields of the standard form. You may choose the order as long as it is very different.
  - ❏ **Different submit action**. Pressing the submit button must cause the controller action, *produce*, to fire (see below).

- ❏ **Add a *produce* action**: Add to the controller associated with your task's model class a *produce* action. This action should be the same as the *create* action, but with the following differences.
  - ❏ **Route to *produce* action**: The route to the *produce* action should be as follows (assuming that your model class is called *WildPig*).
    - ❏ HTTP Method: POST
    - ❏ URI Pattern: /wild_pigs/inventory
  - ❏ **Tied to *inventory* page**: The outcome of saving a model object in the database should be as follows:
    - ❏ **On success**: Send an HTTP redirect to the *inventory* URL.
    - ❏ **On failure**: Render the *inventory* page, such that form-error messages render in the usual way.
- ❏ **Update the *home* page**: Update your hyperlink on the *home* page (recall from Homework 3) so that it now links to the *inventory* page you created.

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 hw5v1 -m 'Tagged Homework 5 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., hw5v2, hw5v3, hw5v4, etc.).