Homework 6: MVC Model Associations

For this homework, you will further refine your team's web app by adding new model classes and adding associations between some of the model classes. Additionally, 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

Similar to previous homeworks, there will be four tasks (Tasks 1 through 4); however, this time the tasks are more inter-related. Each task will have the same five parts:

A. Add a new model class (with scaffold, validations, and tests).
B. Add two model associations.
C. Add seed data that uses the associations.
D. Update views/controllers to make use of the associations.
E. Update the *home* page to link to your newly created pages.

Part A: Add a new model class

Figure 1 (below) depicts a class diagram with the new classes to be added. Each new class is labeled with the number of the task responsible for creating it. For example, the student doing Task 1 must create the Journal class; the student doing Task 2 must create the Article class; etc. The Rails model classes created must match the class diagram exactly (as in Homework 4). Also, generate the scaffold controller/views for each model class that you create (also as in Homework 4).

Add one validation per attribute of your newly created class. Also, write one model test for each attribute that shows that the attribute's validation can catch an invalid value. The choice of validations and tests is largely up to you, but choose something sensible (i.e., not too weird).

Once you have created your new model class, you will likely want to commit and push it as soon as possible. In the next part, other students may need to modify your classes to complete their tasks, so you should do your utmost not to hold them up.

Part B: Add two model associations

Implement two one-to-many associations, as shown in Figure 1 (see below). You must create the associations labeled with your task number. Your Rails model associations must match the class diagram exactly—including the role names on the association ends.

Part C: Add seed data

Add seed data as follows. You must have at least three records from the "one" side of each of your associations, and each of those records must be associated with at least two records from the "many" side of the association. Thus, for a given association, there will need to be a minimum of 9 records (3 from the one side + 2 + 2 + 2 from the many side).

In creating the seed data, you may need to coordinate with your teammates to come up with sensible data given their constraints. To keep the total number of seed objects manageable, it is OK for you to "share" model objects with teammates in the seeds file. That is, you may count the same model object toward the objects required for multiple associations.

Part D: Update views/controllers

Using your newly created associations, you must update views/controllers as per the task-specific requirements below:

- Task 1:
    - Update the TalentAgent *show* page such that it includes a table listing all the Actor objects that the TalentAgent object has. The table should be inserted below the usual TalentAgent *show* info and be styled like the table in the Actor *index* page.
    - Update the Actor *index* page such that a new column is added to the table. The column should be titled "Talent Agent", and it should display the *last_name* attribute of the TalentAgent object to which the Actor belongs.
    - The Actor *new*/*create* and *edit*/*update* pages should now include a dropdown field that allows the user to select the TalentAgent object to which the Actor object will belong. The text for each item in the dropdown should be the *last_name* attribute of the TalentAgent.
- Task 2:
    - Update the Producer *show* page such that it includes a table listing all the Film objects that the Producer object has. The table should be inserted below the usual Producer *show* info and be styled like the table in the Film *index* page.
    - Update the Film *index* page such that a new column is added to the table. The column should be titled "Producer", and it should display the *name* attribute of the Producer object to which the Film belongs.
    - The Film *new*/*create* and *edit*/*update* pages should now include a dropdown field that allows the user to select the Producer object to which the Film object will belong. The text for each item in the dropdown should be the *name* attribute of the Producer.
- Task 3:
    - Update the Reviewer *show* page such that it includes a table listing all the Review objects that the Reviewer object has. The table should be inserted below the usual Reviewer *show* info and be styled like the table in the Review *index* page.
    - Update the Review *index* page such that a new column is added to the table. The column should be titled "Reviewer", and it should display the *handle* attribute of the Reviewer object to which the Review belongs.

    - The Review *new*/*create* and *edit*/*update* pages should now include a dropdown field that allows the user to select the Reviewer object to which the Review object will belong. The text for each item in the dropdown should be the *handle* attribute of the Reviewer.
- Task 4:
    - Update the UserProfile *show* page such that it includes a table listing all the Comment objects that the UserProfile object has. The table should be inserted below the usual UserProfile *show* info and be styled like the table in the Comment *index* page.
    - Update the Comment *index* page such that a new column is added to the table. The column should be titled "Author", and it should display the *name* attribute of the UserProfile object to which the Comment belongs.
    - The Comment *new*/*create* and *edit*/*update* pages should now include a dropdown field that allows the user to select the UserProfile object to which the Comment object will belong. The text for each item in the dropdown should be the *name* attribute of the UserProfile.

Part E: Update the *home* page

Update your hyperlink on the *home* page so that it now has your name (no hyperlink) followed by the following two links:

- Task 1:
    - "Actors" linking to Actor *index* page.
    - "Talent Agents" linking to TalentAgent *index* page.
- Task 2:
    - "Producers" linking to Producer *index* page.
    - "Films" linking to Film *index* page.
- Task 3:
    - "Reviewers" linking to Reviewer *index* page.
    - "Reviews" linking to Review *index* page.
- Task 4:
    - "Comments" linking to Comment *index* page.
    - "User Profiles" linking to UserProfile *index* page.

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 hw6v1 -m 'Tagged Homework 6 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., hw6v2, hw6v3, hw6v4, etc.).

↓ **Figure Below** ↓

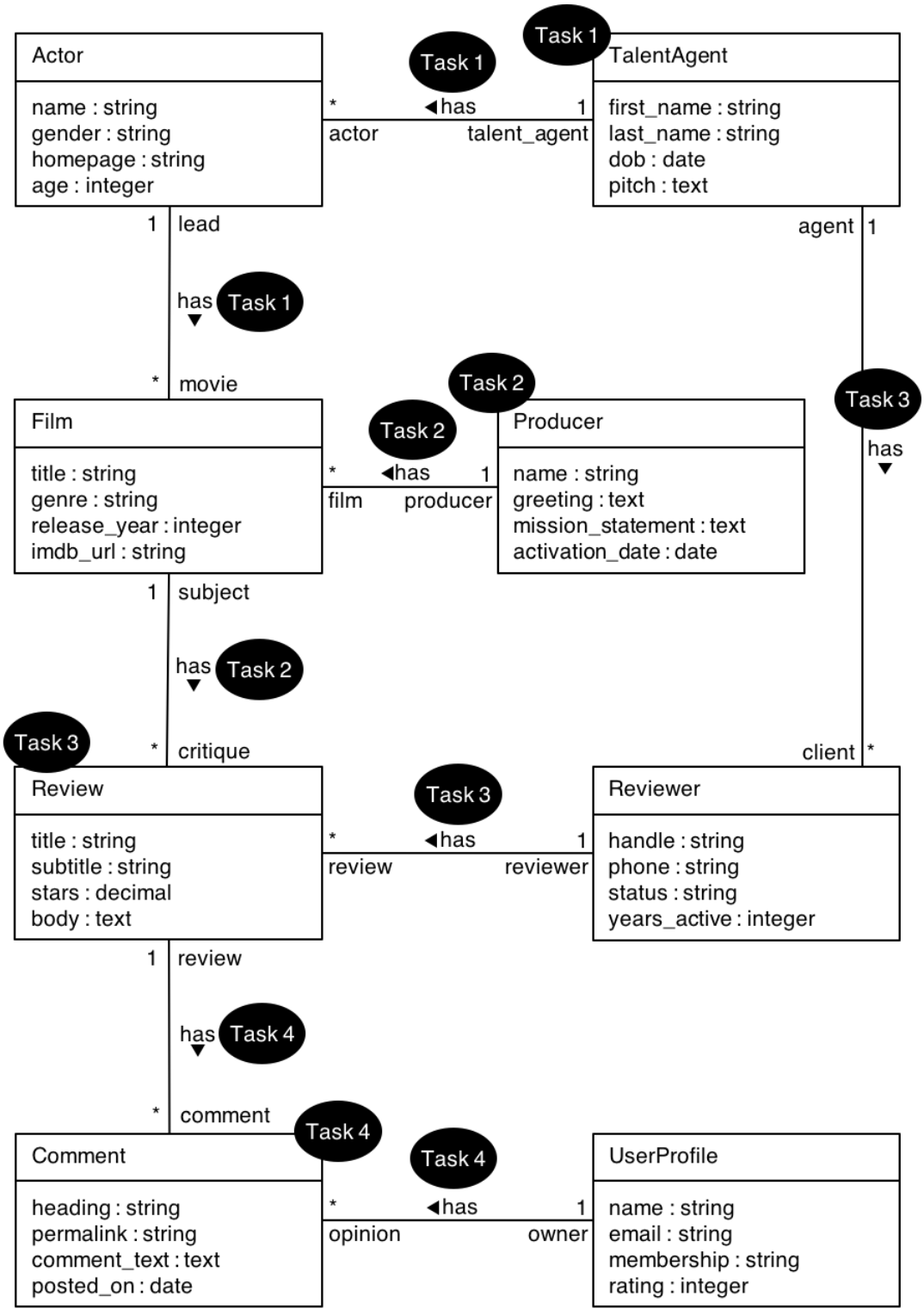

Figure 1. Model class diagram and task assignments.