CS142 Project #4: Migrations and Models

In this project you will use Rails to create the beginnings of a photo-sharing Web site. For this project, the Web site will display pre-entered information in the database. In future projects you will add additional features, such as the ability to add new photos and to tag photos with the names of users.

Setup

Create a Rails application for this project by invoking the following command:

```
rails new project4
```

This will create a directory `project4` that contains a skeleton for the project. Your solutions for the problems below should be implemented in this project. Follow the instructions described under *Additional Setup* for Project #3 to modify the default Rails configuration.

Problem 1: Use Migrations to Create the Database (10 points)

- Photos in the photo-sharing site are organized by user. Start off by creating the model class `User` and its associated database table, with the following attributes:

`id`	Primary key for this user.
`first_name`	First name of the user.
`last_name`	Last name of the user.

  To do this, change to the `project4` directory and invoke the following command:

  ```
  rails generate model user
  ```

  This will create a migration file with a name something like `20120330202846_create_users.rb` in the directory `db/migrate`. It will also create an initial model file `app/models/user.rb`. Edit the migration file, remove the method `change` that is created by default, and replace it with separate methods `up` and `down` as described in class (`change` is an alternate way of writing migrations introduced in Rails 3.0; it's more compact than the up/down approach, but it's a little less obvious, so we won't use it in this class).

- Each user can upload multiple photos. Use the same approach as above to create the model `Photo` and its associated database table, with the following attributes:

`id`	Primary key for this photo.
`user_id`	Identifier for the user who created the photo.
`date_time`	The date and time when the photo was added to the database.
`file_name`	Name of a file containing the actual photo (in the directory `project4/public/images`).

- For each photo there can be multiple comments (any user can comment on any photo). Use the same approach as above to create the model `Comment` and its associated database table, with the following attributes:

`id`	Primary key for this comment.
`photo_id`	Identifier for the photo to which this comment belongs.
`user_id`	Identifier for the user who created the comment.
`date_time`	The date and time when the comment was created.
`comment`	The text of the comment.

- Modify the model classes created above to include the relationships between models (e.g. each photo is associated with a user, and each comment is associated with both a photo and the user who created the comment).

- Create an additional *data migration* that will pre-load the database with some initial data. To do this, invoke the following command:

  ```
  rails generate migration load_data
  ```

Then find the new migration file and replace its contents with the contents of this file: load_data.rb. In addition, copy all of the images from this zip file into the directory `project4/public/images`.

- Invoke the following command, which will run all of your migrations, creating the database schema and loading the initial data:

```
rake db:migrate
```

Problem 2: Create the Application (10 points)

Create controllers and views that implement two URLs. The first URL is `/users/index`. When this URL is referenced your application must return a Web page that displays a list of all users in the database. The user names must be links: clicking on a user name displays the photos for that user, as described below.

The second URL supported by your application has the form `/photos/index/`*id*, where *id* is the database identifier for a particular user. When this URL is referenced your application must return a web page displaying all of the photos belonging to that user. For each photo you must display the photo itself, the creation time for the photo, and all of the comments for that photo. For each comment you must display the time when the comment was created, the name of the user who created the comment, and the text of the comment. The creator for each comment should be a link that can be clicked to switch to the photos page for that user.

Although you don't need to spend a lot of time on the appearance of the pages you generate, they should be neat and understandable. They should have meaningful titles, and the information layout should be clean (e.g., it should be clear which photo each comment applies to). Use the personalized layout you created for the last project.

Style Points (5 points)

These points will be awarded if your problem solutions have proper MVC decomposition, follow the Rails conventions, and generate valid XHTML. In addition, your code and templates must be clean and readable, and your Web pages must be at least "reasonably nice" in appearance and convenience.

Deliverables

Use the standard class submission mechanism to submit the entire application (everything in the `project4` directory). As usual, your generated HTML must validate as proper XHTML (you do not need to validate your CSS for this project).