Assignment for Module #1: Basic Active Record CRUD

The overall goal of this assignment is to assess your ability to implement basic CRUD functionality of Active Record. This includes:

- Creating Active Record Models using a rails-provided generator (rails g model or rails g scaffold)
- Creating a DataBase (DB) schema
- Inserting rows in the DB
- Updating rows in the DB
- Querying the DB with exact matching
- Getting rows from the DB by Primary Key (PK)
- Deleting rows from the DB
- Retrieving paginated results from the DB using limit and offset keywords

This does NOT include:

- Seeding with seeds.rb
- Relationships
- Validations
- Advanced queries

These topics of Active Record are covered in Module 2.

The functional goal of this assignment is to implement CRUD behavior for four (4) Model classes

1. User
2. Profile
3. TodoList
4. TodoItem

Functional Requirements

1. Create several Active Record Model classes
    - User
    - Profile
    - TodoList
    - TodoItem
2. Create a DB schema for each Model class
3. Demonstrate CRUD access to information in the DB using Active Record Model methods.

Getting Started

1. Create a new Rails application called todolists.
2. Add the following specification to your Gemfile to enable rspec testing.

```
group :test do
  gem 'rspec-rails', '~> 3.0'
end
```

3. Run the bundle command to resolve new gems
4. From the todolists application root directory, initialize the rspec tests using rails generate rspec:install command.

```
[todolists]$ rails generate rspec:install
create  .rspec
create  spec
create  spec/spec_helper.rb
create  spec/rails_helper.rb
```

   Add the following line to .rspec to add verbose output to test results.

```
--format documentation
```

5. Download and extract the starter set of boostrap files.

```
|-- Gemfile
|-- assignment
|   `-- assignment.rb
`-- spec
    `-- assignment_spec.rb
```

   - overwrite your existing Gemfile with the Gemfile from the bootstrap fileset. They should be nearly identical, but this is done to make sure the gems and versions you use in your solution can be processed by the automated Grader when you submit. Any submission should be tested with this version of the file.
   - add the assignment/assignment.rb file provided with the boostrap fileset to a corresponding assignment directory under your application root directory (e.g., application-root-directory/assignment/assignment.rb). (Note: You will need to create the assignment directory if you are copying the file.) The assignment.rb file contains a skeleton of methods for you to implement as part of your assignment.
   - add the spec/assignment_spec.rb file provided with the bootstrap fileset to the corresponding spec directory that already exists under your application root directory (e.g., application-root-directory/spec/assignment_spec.rb). This assignment_spec.rb file contains tests that will help determine whether you have completed the assignment.

6. Run the rspec test(s) to receive feedback. rspec must be run from the root directory of your application. All tests will (obviously) fail until you complete the specified solution.

```
$ rspec
...
Finished in 0.02069 seconds (files took 1.63 seconds to load)
52 examples, 1 failure, 50 pending
```

   To focus test feedback on a specific step of the requirements, add "-e rq##" to the rspec command line to only evaluate that requirement. Pad all step numbers to two digits.

```
$ rspec -e rq01
Run options: include {:full_description=>/rq01/}

Assignment
  rq01
    Generate Rails application
      must have top level structure of a rails application

Finished in 0.00465 seconds (files took 1.56 seconds to load)
1 example, 0 failures
```

7. Implement your Model and assignment.rb solution and use the rspec tests to help verify your completed solution.
8. Submit your Rails app solution for grading.

Technical Requirements

1. Create a new Rails app called todolists.

```
$ rspec -e rq01
```

   Note: Use the Gemfile provided in the boostrap files. Windows users may need to add this gem into the Gemfile:

```
# Windows does not include zoneinfo files, so bundle the tzinfo-data gem
gem 'tzinfo-data', platforms: [:mingw, :mswin]
```

   Otherwise, do not change the Gemfile from what is provided or your submitted solution may not be able to be processed by the grader (i.e., do not add any additional gems or change gem versions).

2. Generate four (4) model classes and DB migrations having the following business-related fields using one of the rails generate commands.

   You can individually grade your results after each model class is created by migrating the database and running the rspec test listed after each classname. Example:

   ```
   $ rake db:migrate
   $ rspec -e rq02.1
   ```

   1. User (rq02.1)
      - username - a string to hold account identity
      - password_digest - a string to hold password information
   2. Profile (rq02.2)
      - gender - a string to hold the words "male" or "female"
      - birth_year - a number to hold the year the individual was born
      - first_name - a string with given name of user
      - last_name - a string with family name of user
   3. TodoList (rq02.3)
      - list_name - a string name assigned to the list
      - list_due_date - a date when TODO items in the list are to be complete. This is a date. We are not concerned with the time of day.
   4. TodoItem (rq02.4)
      - due_date - date when the specific task is to be complete
      - title - a string with short name for specific task
      - description - a text field with narrative text for specific task
      - completed - a boolean value (default=false), indicating whether item is complete

   Reminder: Ruby/Rails conventions are that class names are CamelCase and file and method names are snake_case.

   Note: We will only be using the Model and DB migration classes for this assignment. It is assumed that each Model will also contain the id, created_at, and updated_at fields. All four (4) Model Classes/DB tables must be created, but the detailed tests will be focused on the User and TodoList only. You will use the remaining classes more in a follow-on assignment.

3. Insert rows in DB
   1. Implement the create_user method within assignment/assignment.rb to
      - accept a hash of user properties (:username and :password_digest) as an input parameter. Note these are 100% same as model class.
      - use the User Model class to create a new user in the DB
      - return an instance of the class with primary key (id), and dates (created_at and updated_at) assigned

      ```
      $ rspec -e rq03.1
      ```

   2. Implement the create_todolist method within assignment/assignment.rb to
      - accept a hash of todolist properties (:name and :due_date) as an input parameter. **Note** these hash keys are not 100% the same as Model class. Your solution must convert these properties into hash with keys Active Record will accept for the schema for the class.
      - use the TodoList Model class to create a new todolist in the DB
      - return an instance of the class with primary key (id), and dates (created_at and updated_at) assigned

      ```
      $ rspec -e rq03.2
      ```

4. Retrieve paginated results from DB
   1. Implement the find_allusers method within assignment/assignment.rb to

- accept offset and limit input parameters
- use the User Model class to find all Users, ordered by updated_at ascending, with specified row offset and row limit
- return a collection of User instances that represent the specified page

```
$ rspec -e rq04.1
```

2. Implement the find_alllists method within assignment/assignment.rb to

    - accept offset and limit input parameters
    - use the TodoList Model class to find all TodoLists, ordered by list_due_date descending, with specified row offset and row limit
    - return a collection of TodoList instances that represent the specified page

    ```
    $ rspec -e rq04.2
    ```

5. Query DB with exact match

    1. Implement the find_user_byname method within assignment/assignment.rb to

        - accept a username input parameter
        - use the User Model class to find all Users with the supplied username. Note that we have not yet constrained the username to be unique.
        - return a collection of User instances that match the provided username

        ```
        $ rspec -e rq05.1
        ```

    2. Implement the find_todolist_byname method within assignment/assignment.rb to

        - accept a name input parameter
        - use the TodoList Model class to find all TodoLists with the supplied list_name. Note that list_name is not required to be unique.
        - return a collection of TodoList instances that match the provided name

        ```
        $ rspec -e rq05.2
        ```

6. Get rows from DB by PK

    1. Implement the get_user_byid method within assignment/assignment.rb to

        - accept an id input parameter
        - use the User Model class to get the User associated with the id primary key
        - return the User instance that matches the provided id

        ```
        $ rspec -e rq06.1
        ```

    2. Implement the get_todolist_byid method within assignment/assignment.rb to

        - accept an id input parameter
        - use the TodoList Model class to get the TodoList associated with the id primary key
        - return the TodoList instance that matches the provided id

        ```
        $ rspec -e rq06.2
        ```

7. Update rows in DB

    1. Implement the update_password method within assignment/assignment.rb to

        - accept an id and password_digest input parameters
        - use the User Model class to update the password_digest for the User associated with the id primary key
        - (no return is required)

        ```
        $ rspec -e rq07.1
        ```

    2. Implement the update_listname method within assignment/assignment.rb to

- accept an id and name input parameters
- use the TodoList Model class to update the list_name for the TodoList associated with id primary key
- (no return is required)

```
$ rspec -e rq07.2
```

8. Delete rows from DB

    1. Implement the delete_user method within assignment/assignment.rb to

        - accept an id input parameter
        - use the User Model class to delete the User associated with the id primary key from the database
        - (no return is required)

        ```
        $ rspec -e rq08.1
        ```

    2. Implement the delete_todolist method within assignment/assignment.rb to

        - accept an id input parameter
        - use the TodoList Model class to delete the TodoList associated with the id primary key.
        - (no return is required)

        ```
        $ rspec -e rq08.2
        ```

Self Grading/Feedback

Some unit tests have been provided in the bootstrap files and provide examples of tests the grader will be evaluating for when you submit your solution. They must be run from the project root directory.

```
$ rspec
...
Finished in 3.39 seconds (files took 1.47 seconds to load)
52 examples, 0 failures
```

You can run as many specific tests you wish be adding -e rq## -e rq##

```
$ rspec -e rq01 -e rq02
```

Submission

Submit an .zip archive (other archive forms not currently supported) with your solution root directory as the top-level (e.g., your Gemfile and sibling files must be in the root of the archive and *not* in a sub-folder. The grader will replace the spec files with fresh copies and will perform a test with different query terms.

```
|-- app
|   |-- assets
|   |-- controllers
|   |-- helpers
|   |-- mailers
|   |-- models
|   `-- views
|-- assignment
|   `-- assignment.rb
|-- bin
|-- config
|-- config.ru
|-- db
|-- Gemfile
|-- Gemfile.lock
|-- lib
|-- log
|-- public
|-- Rakefile
```

```
|-- README.rdoc
|-- test
`-- vendor
```

Last Updated: 2015-10-29