A simple CRUD application using Sinatra and Bootstrap

by Cerize Santos

The application allows the user to:

- Create new tasks
- Mark a task as complete or incomplete
- Delete a task
- Edit a task

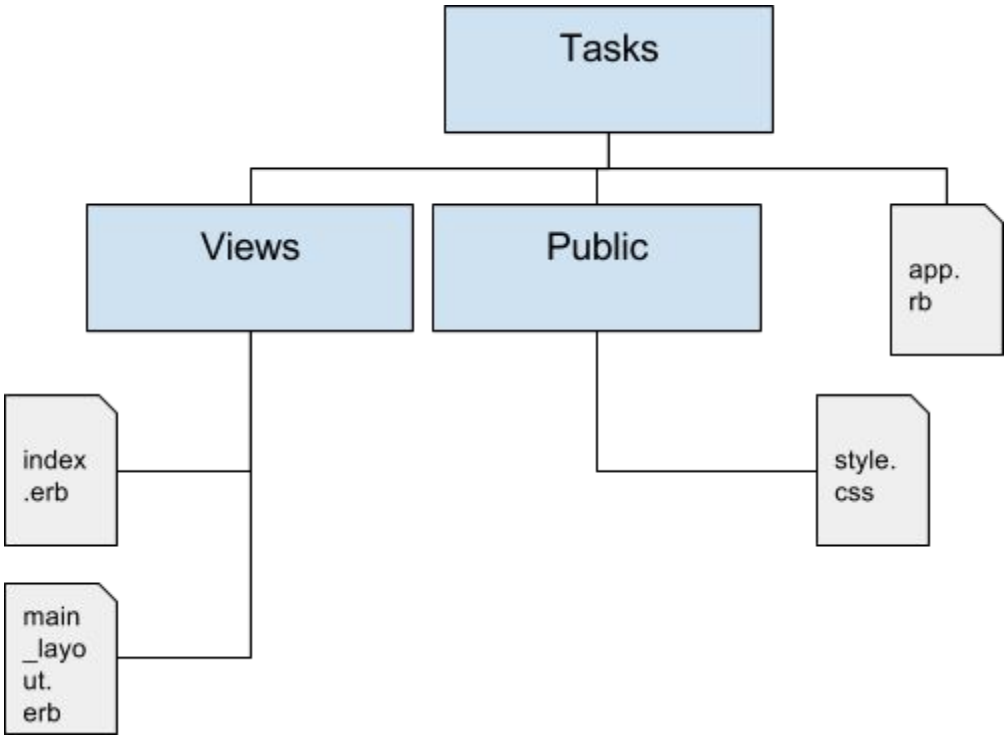
Database:

- SQLite

Gems:

- sinatra
- sinatra-contrib (for sinatra/reloader)
- data_mapper (connects with a database and allows an object oriented approach for manipulating the data)
- dm-sqlite-adapter (to connect data_mapper with SQLite)

1. **Create the basic folder structure**: tasks (main folder), views and public.
2. **Create your basic files**: app.rb, index.erb, main_layout.erb, style.css. Your file system should look like this:



3. **Check if everything is working so far**. On your app.rb:

```
require 'sinatra'
require 'sinatra/reloader'
require 'data_mapper'

get '/' do
      "Hello world!"
end

# run app.rb on the terminal and check the result at http://localhost:4567/
```

4. **Work on your main.layout.erb**:

- create your basic html structure;
- add bootstrap and style.css links:

```
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></s
cript>
```

```
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"
integrity="sha384-0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHV
P9aJ7xS" crossorigin="anonymous"></script>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css
"
integrity="sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjW
PGmkzs7" crossorigin="anonymous">
    <link rel="stylesheet" href="style.css">
```

- Indicate where the content of other .erb pages should be inserted with `<%=yield%>`

5. **Put some content on index.erb and test it by editing the get method**:

```
get '/' do
      erb :index, layout: :main_layout
end
```

6. **Put some style on style.css and test it**:

```
body {
      font-family: 'Ubuntu', sans-serif;
      font-size: 18px;
      color: #66595C;
}
```

7. **Now it's time to create your database**:

```
DataMapper.setup(:default, "sqlite://#{Dir.pwd}/tasks.db")

class Task
      include DataMapper::Resource
      property :id, Serial
      property :content, String
      property :complete, Boolean, :default  => false
end

DataMapper.finalize.auto_upgrade!
```

- The resulting database will have a name “tasks” and a table called “tasks” (that comes from the plural of the class name.

- The table will have 3 columns, one auto incremented id, one called ‘content’, which is a String (default length limit of 50 characters), and one “complete”, which indicates if the task was completed or not. It’s set as `false` as default when the user creates a new task.
- If some property were set as Text, it defaults to lazy loading and length limit of 65535 characters. You have to `:lazy => false` to update/show a Text data type: by default DataMapper will not load it from database when doing `ClassName.all`.
- If you refresh http://localhost:4567/, you should see the creation of the file tasks.db inside your main folder.

8. **On index.erb, create the basic form to interact with the user, receiving input** (create new tasks):

```
<form action="/" method="post">
  <input type="text" name="content" >
  <input type="submit" value="Create a task">
</form>
```

- remember: the instance variables used on any .erb file must be defined on app.rb on the appropriate method.

9. **On app.rb, create the post '/' method associated with the form above**:

```
post '/' do
      Task.create(params)
      redirect '/'
end
```

- Insert some data to see if it was all saved on the database (use SQLite Browser).

10. The user must be able to see the tasks he created. Task.all gives an array which elements are records on the database. Create two arrays, one for the completed tasks (:complete => true) and another for incomplete tasks (:complete => false). On index.erb, iterate over each array to show each task:

```
get '/' do
  @tasks_complete = Task.all(:complete => true)
  @tasks_incomplete = Task.all(:complete => false)
  erb :index, layout: :main_layout
end
```

```
<h2>Pending Tasks</h2>
     <%@tasks_incomplete.each do |task|%>
        <p><%=task.content%></p>
    <%end%>

<h2>Complete Tasks</h2>
    <%@tasks_complete.each do |task|%>
        <p><%=task.content%></p>
    <%end%>
```

11. **The user must be able to change the status of a task**.

- on index.erb:

```
  <form action="/complete/<%=task.id%>" method="post">
    <input type="submit" value="Mark as Complete">
  </form>
```

- on app.rb:

```
post '/complete/:id' do
  t = Task.get params[:id]
  t.complete = t.complete ? false : true
  t.save
  redirect '/'
end
```

12. **To delete a task, let's use a modal to confirm**:

- on index.erb:

```
  <!-- Trigger the delete modal-->
  <button type="button" class="btn btn-danger btn-lg" data-toggle="modal"
data-target="#del<%=task.id%>">Delete</button>
<!-- Modal to delete -->
<%@tasks.each do |task|%>
  <!-- Modal -->
  <div id="del<%=task.id%>" class="modal fade" role="dialog">
    <div class="modal-dialog">

      <!-- Modal content-->
      <div class="modal-content">
        <div class="modal-header">
```

```
          <button type="button" class="close"
data-dismiss="modal">&times;</button>
          <h4 class="modal-title">Delete a task</h4>
        </div>
        <div class="modal-body">
          <p>Are you sure you want to delete this task?</p>
        </div>
        <div class="modal-footer">
          <form action="/delete/<%=task.id%>" method="post">
            <input type="hidden" name="_method" value="delete">
            <input type="submit" value="Yes" class="bnt btn-danger">
          </form>
          <button type="button" class="btn btn-default"
data-dismiss="modal">No</button>
        </div>
      </div>

    </div>
  </div>

<%end%>
```

- on app.rb:

```
delete '/delete/:id' do
  t = Task.get params[:id]
  t.destroy
  redirect '/'
end
```

13. To edit a task, let's also use a modal

- on index.erb

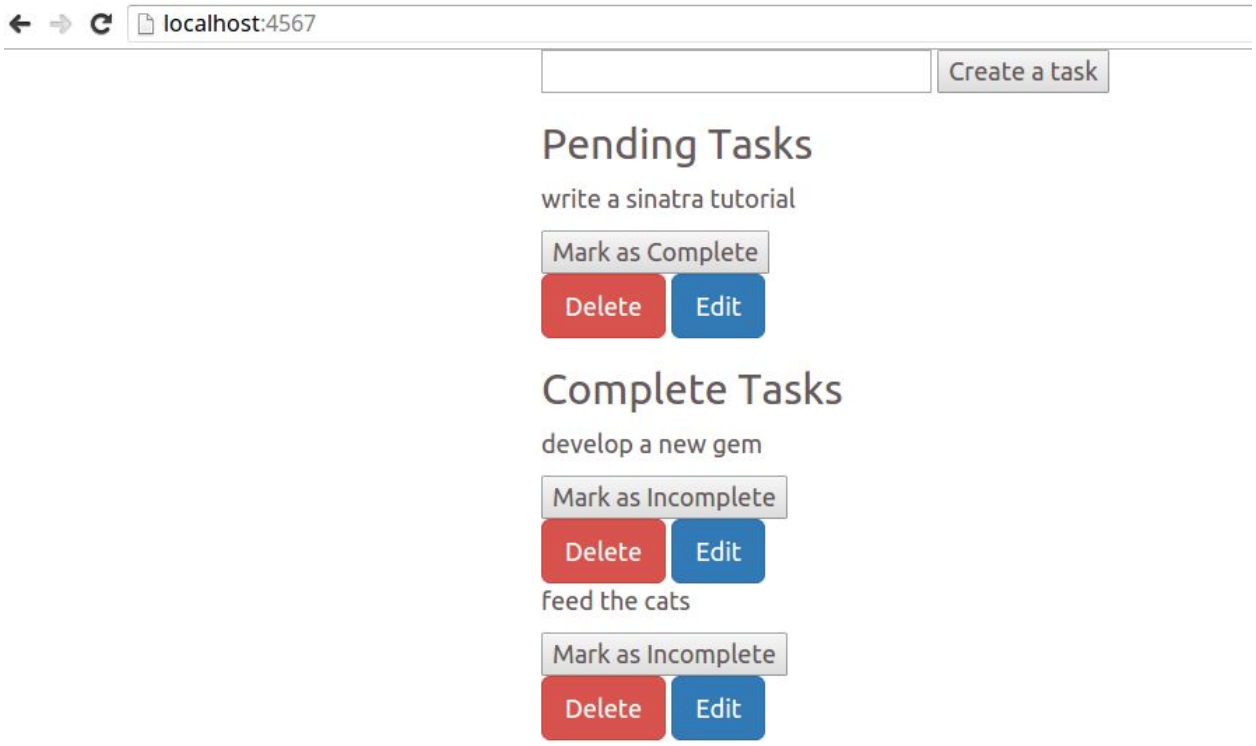
```
<!-- Modal to edit -->
<%@tasks.each do |task|%>
  <!-- Modal -->
  <div id="edit<%=task.id%>" class="modal fade" role="dialog">
    <div class="modal-dialog">
```

```
      <!-- Modal content-->
      <div class="modal-content">
        <div class="modal-header">
          <button type="button" class="close"
data-dismiss="modal">&times;</button>
          <h4 class="modal-title">Edit</h4>
        </div>
        <div class="modal-body">
          <form action="/edit/<%=task.id%>" method="post">
            <input type="hidden" name="_method" value="put">
            <input type="text" value="<%=task.content%>" name="content">
            <input type="submit" value="Save" >
          </form>
        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-default"
data-dismiss="modal">Close</button>
        </div>
      </div>

    </div>
  </div>

<%end%>
```

- on app.rb

```
put '/edit/:id' do
  t = Task.get params[:id]
  t.content = params[:content]
  t.save
  redirect '/'
end
```

14. Great! The app now allows the user to create, read, delete and edit a task. It looks like this:



15. After improvements in the layout:

16. next step: authentication