**Sinatra TodoList: Part 2**

At this point, we should have well tested, well organized TodoList. If you don't, make sure to go back to day 1 and finish up any iterations(besides the 8th, which is optional) you didn't get to before continuing.

What good is our TodoList if it's just a bunch of class and test files? It's cool, but I want to put it on the web.

**Setup**

Navigate to the folder with your todo list stuff in it. Right now, you should have a lib folder with your todo\_list and task class files, a spec folder with the tests for each of those, a public folder with your saved task yaml files. If you haven't already, make sure that any gems that you use are in your Gemfile.

It's time to add sinatra, and sinatra-contrib(for the sinatra reloader). Create a server.rb file that is going to handle the routes for our application.

*#Inside of server.rb*

**require** 'sinatra'

**require** 'sinatra/reloader'

*# We're going to need to require our class files*

require\_relative('lib/Task.rb')

require\_relative('lib/TodoList.rb')

Finally, create a views directory, remember any of our .erb templates are going to go there.

**Iteration 1**

Let's display all of our tasks. First we have to create a "global" todo\_list for all of our actions to access. This isn't the best way to do things, but it will be fine for now.

*#Inside of server.rb*

require 'sinatra'

require 'sinatra/reloader'

*# We're going to need to require our class files*

require\_relative('lib/Task.rb')

require\_relative('lib/TodoList.rb')

todo\_list = TodoList.**new**("Josh")

todo\_list.load\_tasks

Next, create a route for checking out the todo list. The home path should say, "Welcome to the TodoList" and display all of your tasks.

**Route**: '/tasks'(GET)  
**ERB**: task\_index.erb   
**What does it do?**: Displays all of the tasks

**Iteration 2**

Let's add new tasks.

Make a link on your tasks\_index page to "Add New Task". This link should route to/new\_task. This route is responsible for displaying a form, to gather (=recoger) information from our user.

What does this form do you ask? It posts to a route called /create\_task.

create\_task should do 4 things

* Create a new task(Task.**new**)
* Add it to the todo\_list (with the todo\_list.add\_task method)
* Save the todo\_list (with the todo\_list.save method)
* Redirect the user back to the /tasks route

**Route**: '/new\_task'(GET)  
**ERB**: new\_task.erb  
**What does it do?:** Displays a form, which posts to /create\_task

**Route**: '/create\_task'(POST)  
**ERB**: No erb, there's nothing to display. We just want to perform some actions on the back-end, and redirect our user.  
**What does it do?**: This should take the information received from a form, create a new task, add it to the list, and save the tasks. Finally, it should redirect the user back to the tasks\_index. Remember, we want to **redirect**, not render.

**Iteration 3**

Mark (=marcar) tasks as complete. We need to be able to complete our task, so create a link or and have that link to a route called /complete.

Well, kind of (=un poco). We're missing a small step here. When we click the complete link, we only want to complete that **one** task. How are we going to uniquely identify the one specific task that we clicked on? Hmm...remember back on day 1 when I had mentioned we were going to need a way to uniquely identify a task?

This route should find a task from the todo\_list, mark it as complete, and then redirect the user to our list of tasks.

When we return to the list of tasks, there should be some kind of indication that the task is complete. Maybe some text next to it saying complete, be creative, just indicate to your user that the task is complete.

**Route**: '/complete\_task/:id'(GET)  
**ERB**: None  
**What does it do?**:

* This action should find a task out of your todo\_list based on it's id.(todo\_list.find\_task\_by\_id)
* Then, we should mark the task as complete(task.complete!)
* Save the todo list(todo\_list.save)
* Redirect the user back to the task home page.

**Iteration 4**

We need to delete tasks now. In your tasks\_index view, add an additional button to the tasks that are complete. This button should say "Delete".

Deleting is very similar to completing. We're going to need a very similar route that picks out one task through the URL parameters, and deletes it from the todolist.

**Route**: '/delete\_task/'' (POST)   
**ERB**: None  
**What does it do?**:

* Deletes one specific task from the todolist with the todo\_list.delete\_task method
* Save the todo list(todo\_list.save method)
* Redirects the user back to the task home page