Web Design with Kraken

In this lab, you will create a simple four-page application using a newly generated Kraken application.

# Objectives

You will learn how to

* Create the four pages
* Add the EL for the welcome message
* Create a link with a route to the next page

# Install Tools

If you have already created a Kraken app from a previous lab, you may skip this step.

The first thing we have to do is to use Node Package Manager, npm, to install generator-kraken, yo, and bower Do this by opening a command prompt in the lesson directory and issuing the command npm install –g generator-kraken yo bower.

# Lab Steps

Now that the tools are installed, we can begin the lab, whose goal is to scaffold a Kraken web application, add a few pages and link them to the corresponding controllers.

## Scaffold a new web application with Kraken

1. Open a terminal in the lab directory, then issue the command yo kraken. Enter store for the app name, My Store for the description, and whatever you want for the author. Take the defaults for the remaining prompts.

This will scaffold a web application that uses Express and Kraken in the directory store.

## Start & view the web application

1. Change into the store directory and issue the command npm start; this will start the server process.
2. Open http://localhost:8000 in a web browser. You should see something similar to the following:



Once you see this, move on to the next step.

## Create the four pages

1. Use the command line, yo kraken:controller welcome, to create the welcome page.

Use similar commands to create the cart, store, and checkout controllers and pages.

1. Modify the pages to look similar to the following. Each page should link to the controller for the next page.

{>"layouts/master" /}  
  
{<body}  
 <h1>{@pre type="content" key="greetings"/}</h1>  
 <h2>{welcome}</h2>  
 <a href=”/store”>Go To The Store</a>   
{/body}

## Edit the controllers

1. Add the model.welcome property to include the name of the next page. The controller should already use res.render(“page name”, model) to render the next page.
2. Stop the web server (Ctrl-C) and restart it with npm start.
3. Open the browser using <http://localhost:8000/welcome> Test each page to ensure that the welcome message is correct and that the correct controller is called.
4. Now that the navigation and topology runs correctly, it is time to add the real application and view code. This is left as an exercise for the user ☺.
5. I usually create a services layer that returns fake data and work on the controllers and pages. The services layer isolates the controller logic from the business logic. Then I work on the services layer to return actual data for the pages. Actually, I iterate around all three concerns as the project continues. Have fun!!