AngularJS SPAs Lab

Let's say we've noticed that the productSearch page, the productList page, and the productDetail page all have some things in common; they deal with products and they have the same layout. We can take advantage of some efficiencies by combining all of these into a single page.

Converting the shell page

One of the three needs to be the shell page. We'll arbitrarily pick productList.html.

1. Copy productList.html to index.html. (Hint: Make sure you're in your product subdirectory so you don't clobber your real index.html!)
2. Open this index.html and add this script tag to the document's <head>:

```
<script src="/app/node-modules/angular-route/angular-route.js">
</script>
```

3. Locate the content section. (Hint: That's the stuff that is different from the other two pages.)
4. Delete all of the content section. Make sure to leave the parts that are common to productSearch.html and productDetail.html.
5. Where the content used to be, make it say this:

```
<section ng-view></section>
```

The ng-view is the part that will make this work. If you were to test this out now, it wouldn't work because we haven't told Angular which partial to use in that view. That's next.

6. Rename productList.html to productListPartial.html. Remove all the parts that are already in its shell page (index.html from above).

Setting up the routing

We need routing for a SPA but we haven't covered all the details yet. We'll explain more what all this stuff means later. Routing must be set up in a config.

7. Edit productModule.js. Add a new dependency to ngRoute and add a config. Do something like this in it:

```
angular.module("productModule", ['ngRoute'])
  .config(function ($routeProvider) {
    $routeProvider
      .when('/search', {
        controller: 'productSearchController',
        templateUrl: "/app/product/productSearchPartial.html",
        caseInsensitiveMatch: true
      })
      .when('/browse', {
        controller: 'productListController',
        templateUrl: "/app/product/productListPartial.html",
        caseInsensitiveMatch: true
      })
```

```
    .when('/', {
      controller: 'productDetailController',
      templateUrl: "productDetailPartial.html",
      caseInsensitiveMatch: true
    });
```

8. As a test, you should be able to navigate to <yourSiteRoot>/product/index.html#/browse and see your list of products. Give it a try.

Converting the other pages

Once our browse page is working, the other pages should be pretty simple to convert.

9. Rename productSearch.html to productSearchPartial.html.
10. Delete everything except the unique content just like you did with the productListPartial.html page.
11. Now do the same with productDetail.html.
12. Run and test. You should be able to see both of these pages at <yourSiteRoot>/product/index.html#/search and <yourSiteRoot>/product/index.html#/.

Examining the architecture

13. In your browser, open the developer tools. Focus on the Net/Network tab.
14. Browse to <yourSiteRoot>/product/index.html. Look at the traffic in the tool.
15. Browse to <yourSiteRoot>/product/index.html#/search. Look at the traffic. You should see all the JavaScript, CSS, and HTML files loaded.
16. Now Browse to <yourSiteRoot>/product/index.html#/browse. What was loaded this time?

What you should see is that only the content was refreshed in the browser. This saves us strain on our server, traffic on our wires, UX time in reloading, and will provide a better experience for our user. Everyone wins!