Currently at: Building a Default View (0:25/2:23)

Consider taking the course “AngularJS Patterns: Clean Code”

===

Usual way of doing REST API calls:

$http.get("/api/products/")

   .then(function (response) {

       vm.products = response.data;

   });

A better alternative is to use $resource

function productResource($resource) {

   return $resource("/api/products/:productID");

}

---

By default, the $resource object provides 5 methods:

Get: GET request, expects JSON object in the response

Query: GET request, expects JSON array in the response

Save: POST request

Delete: DELETE request

Remove: Same as delete, often used instead because delete is a keyword in JavaScript

$resource does not include a PUT request. Can extend $resource to include it.

---

ng-init used to define the model directly in the HTML (only use this for quick testing purposes, kind of like inline CSS)

===

You can fake a backend implementation using $httpBackend

Need angular-mocks.js file

Create a new module that depends on ngMockE2E

var app = angular.module(“productResourceMock, [“ngMockE2E”]);

app.run() executed when the module is loaded

Inside of app.run(), define the model, the url you wish to intercept and the $httpBackend action you wish to perform:

(function () {

   "use strict";

   var app = angular

               .module("productResourceMock",

                       ["ngMockE2E"]);

   app.run(function ($httpBackend) {

       var products = [<list of products> ];

       var productUrl = "/api/products"

       $httpBackend.whenGET(productUrl).respond(products);

   })

}());

**Routing**

http://www.mysite.com/.../**#/products**

Angular routing takes advantage of the fact that the fragment identifier (the stuff after the hash) is processed by the client and not submitted to the server (hence no page refresh)

Angular has two commonly used routing frameworks

ngRoute- Distributed as an additional module, angular.route.js

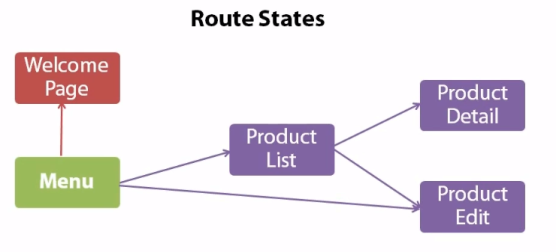
UI Router- Third-party library, more full-featured

ngRoute is based on URL fragment identifiers

UI router is based on application states

Site Map

A site map is a diagram that visually displays how a user can navigate the site. Each page is a box and routes are shown with arrows.



Activating a Route (UI-Router)

3 options

-Set the url

-Call $state.go(<name of state>)

-Click a link with the ui-sref directive ( ex/ <a ui-sref=“…”>Link</a> )

Note that in ui-sref, you want to pass in the name of the state. Ui-sref navigation gets a little more complicated for variable paths:

Normally: <a ui-sref=“productList”>Product List</a>

With a variable path: <a ui-sref=“productEdit({productId:0})”>Add Product</a>