AngularJS

AngularJS Routing

Lesson Objectives

- AngularJS Routing Basics
- Understanding Routing Modes
- Working with $routeParams Service
- Working with $route Service
- Working with $location Service
- Working with Routing Events

Routing

- It is very important to navigate from one page view to another in single page application.
- We can achieve this by including multiple templates in the view using ng-include directive, but this will be unmanageable and also make it difficult to allow other developers to join in the development.
- We can break out the view into a layout and template views and only show the view which we want to show based upon the URL the user is accessing. Routing means loading sub-templates depending upon the URL of the page.
- Routes are a way for multiple views to be used within a single HTML page. This enables you page to look more "app-like" because users are not seeing page reloads happen within the browser.

AngularJS Routes

- **AngularJS routes enable us to create different URLs for different content in our application. Having different URLs for different content enables the user to bookmark URLs to specific content. In AngularJS each such bookmarkable URL is called a route.**
- **AngularJS routes enables us to show different content depending on what route is chosen. A route is specified in the URL after the # sign**
  - http://igate.com/index.html#/training

Setting up page for routing

- **To setup a page for routing we need to follow the 4 steps given below**
- **AngularJS requires the route service, which is not part of the default Angular library. We need to load angular-route.js, as part of your script loading.**
  - <script type="text/javascript" src="Scripts/angular-route.js"></script>
- **We need to inject the route service in our app module**
  - var app = angular.module('routeApp',['ngRoute']);
- **Use ngView directive in the HTML tag(use div tag) to display the given route.**
  - <div ng-view />
- **Configure $routeProvider in the module's config() function via calls to the when() and otherwise() functions.**

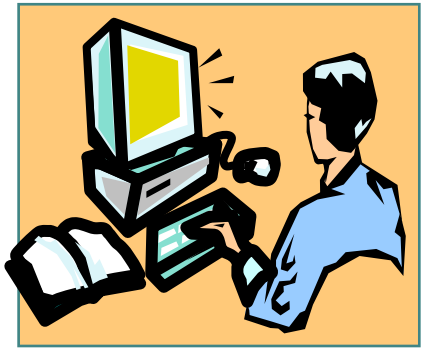
Setting up page for routing

```
var app = angular.module('routeApp',['ngRoute']);
app.config(function($routeProvider){
	$routeProvider
		.when('/',
		{
			template:'<h1>Home Page</h1>'
		})
		.when('/company',
		{
			template:'<h1>IGATE</h1>'
		})
		.otherwise({
			redirectTo:'/'
		})
});
```

- **When the browser loads the Angular app, it will default to the URL set as the default route. Unless we load the browser with a different URL, the default is the '/' route.**

Demo

- **RouteBasics**



Routing Modes

- **Routing mode refers specifically to the format of the URL in the browser address bar. It determines the look of the URL. AngularJS has 2 routing modes**
- **Hashbang Mode**
  - The default behavior of the $location service is to route using the hashbang mode. It provides deep-linking capabilities to Angular apps. URL paths take a prepended '#' character. We can configure hashbang mode in the config function on an app module. We can also configure the hashPrefix, which is part of the fallback mechanism that Angular uses for older browsers.

```
var app = angular.module('routeApp',['ngRoute']);
app.config(function($routeProvider,$locationProvider){
        $locationProvider.html5Mode(false);
        $locationProvider.hashPrefix('!');
});
```

Routing Modes

HTML5 Mode

- This mode makes URLs look like regular URLs (except that in older browsers they will look like the hashbang URL).
- $location service automatically falls back to using hashbang URLs if the browser doesn't support the HTML5 history API and also rewrites the URL.
- For example, with the tag: Employee, a legacy browser's URL will be rewritten to the hashbang URL equivalent: /index.html#! /employee/36?show=true

```
var app = angular.module('routeApp',['ngRoute']);
app.config(function($routeProvider,$locationProvider){
        $locationProvider.html5Mode(true);
});
```

Route Parameters

- **AngularJS will parse a route param with a colon (:) and pass it to $routeParams.**

```
var app = angular.module('routeApp',['ngRoute']);
app.config(function($routeProvider){
        $routeProvider
        .when('/Employees/:id',
        {
                templateUrl:'partials/employees.html',
                controller:'EmployeeController'
        });
});
```

- **Angular will populate the $routeParams with the key of :id, and the value of key will be populated with the value of the loaded URL. If the browser loads the URL /Employees/714709, then the $routeParams object will look like: {id:714709}**

Route Parameters

- **Parameter samples**
  - '/igate' : Matches exactly **igate**
  - '/employee/:id' : Matches employee /714709, employee /desigan
- **We can also specify parameters as query parameters following a '?'**
  - '/employee/:id' : Matches employee /714709?department=training&company=IGATE

```
var app = angular.module('routeApp',['ngRoute']);
app.config(function($routeProvider){
        $routeProvider .when('/employee/:id',
        {
                redirectTo:function(routeParams,path,search){
                    console.log(routeParams);  // Object {id: "714709"}
                     console.log(path);              //   /employee/714709
console.log(search);    // Object {department: "training", company: "IGATE"}
                      return "/";
                }
        })
});
```

$routeParams

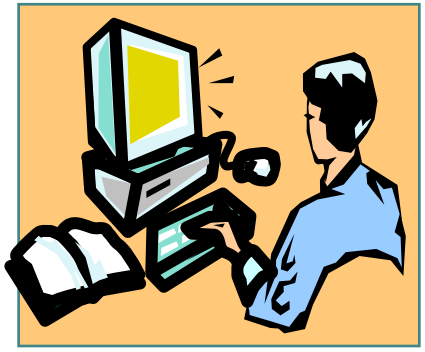
- **The $routeParams service allows you to retrieve the current set of route parameters. Controller functions can get access to route parameters via the AngularJS $routeParams service**

```
var app = angular.module('routeApp',['ngRoute']);
app.config(function($routeProvider){
	$routeProvider
		.when('/:company',
		{
			templateUrl:'partials/map.html',
			controller:'RouteController'
		})
});
app.controller("RouteController",function($scope,$routeParams){
	$scope.model = $routeParams.company;

});
```

Demo

- **RouteParams**

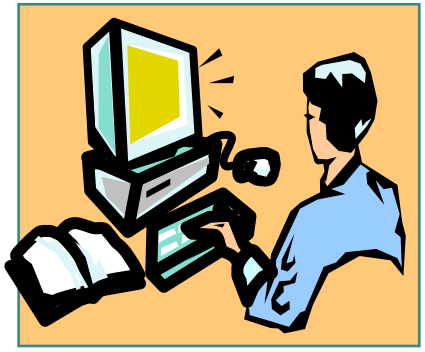


$route

- **$route service is used for deep-linking URLs to controllers and views (HTML partials).**
- **It watches $location.url() and tries to map the path to an existing route definition.**
- **Using route service in the controller, we can access**
  - custom property defined in routing ($route.current.propertyName)
  - URL of the template($route.current.templateUrl)
  - contents of the template($route.current.locals.$template)
  - route parameters($route.current.pathParams)
  - query string passed in URL ($route.current.params)
- **We can reload the partial page using $route.reload()**

Demo

- **RouteService**

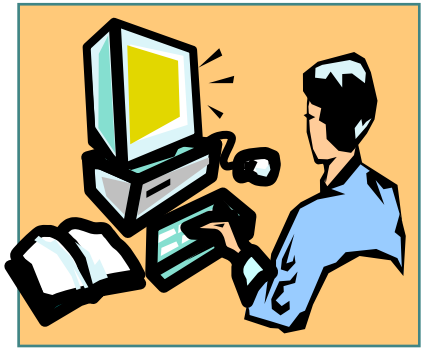


resolve property

- **Resolve is a property on the routing configuration, and each property on resolve can be an injectable function (it can ask for service dependencies). The function should return a promise.**
- **A resolve contains one or more promises that must resolve successfully before the route changes. i.e. We can wait for the data available before showing a view**
- **It simplifies the initialization of the model inside a controller because the initial data is given to the controller instead of the controller needing to go out and fetch the data.**
- **When the promise completes successfully, the resolve property is available to inject into a controller function**

Demo

- **RouteService-Resolve**

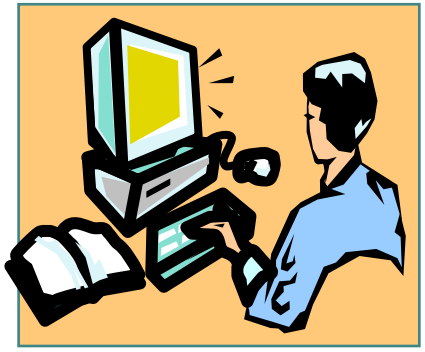


$location service

- **$location service parses the URL in the browser address bar and makes the URL available to the application. The route parameters are a combination of $location's search() and path()**
- **Route service watches $location.url() and tries to map the path to an existing route definition.**
- **$location service is used when application needs to react to a change in the current URL or there is a need to change the current URL in the browser.**

Demo

- **Route-UsingLocationService**



Route Events

- **$route service fires events at different stages of the routing flow. We can set up event listeners for these different routing events and react.**
- **It is useful when we want to manipulate events based upon routes and is particularly useful for detecting when users are logged in and authenticated.**
- **Using $rootScope, we can set up an event listener to listen for routing events.**
- **$routeChangeStart**
  - Angular broadcasts $routeChangeStart before the route changes. This step is where the route services begin to resolve all of the dependencies necessary for the route change to happen and where templates and the resolve keys are resolved.
  - The $routeChangeStart event fires with two parameters:
    - The next URL to which we are attempting to navigate
    - The URL that we are on before the route change

Route Events

- **$routeChangeSuccess**
  - Angular broadcasts the $routeChangeSuccess event after the route dependencies have been resolved.
  - The $routeChangeSuccess event fires with three parameters:
    - The raw Angular evt object
    - The route where the user currently is
    - The previous route (or undefined if the current route is the first route)
- **$routeChangeError**
  - Angular broadcasts the $routeChangeError event if any of the promises are rejected or fail.
  - The $routeChangeError event fires with three parameters:
    - The current route information
    - The previous route information
    - The rejection promise error

Route Events

- **$routeUpdate**
  - Angular broadcasts the $routeUpdate event if the reloadOnSearch property has been set to false and we're reusing the same instance of a controller.

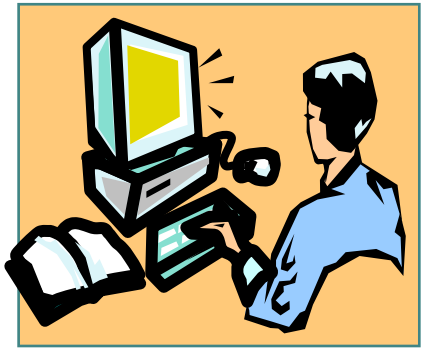
```
angular.module('myApp', [])
.run(['$rootScope', '$location', function($rootScope, $location) {
        $rootScope.$on('$routeChangeStart', function(evt, next, current) {
        })
}])

angular.module('myApp', [])
.run(['$rootScope', '$location', function($rootScope, $location) {
        $rootScope.$on('$routeChangeSuccess', function(evt, next, previous) {
        })
}])

angular.module('myApp', [])
.run(['$rootScope', '$location', function($rootScope, $location) {
    $rootScope.$on('$routeChangeError', function(current, previous, rejection) {
    })
}])
```

Demo

- **Route-RouteEvents**



Summary

- $routeProvider service use to create routes.
- otherwise $routeProvider function allows us to set a default route.
- Using $routeParams service we can access the parameters passed on a route.
- $route.reload() allows us to refresh a view without refreshing the entire app.
- We can enable HTML5 routing using $locationProvider service *$locationProvider.html5Mode(true)*, but it requires server-side configuration.
- resolve route property allows us to delay loading a view until the data which needs to get render is loaded.

Summary

- We can navigate to a new view from the code using $location.url('newUrl').
- $location.search() gives us the access to the query string parameters on the URL.