

Estructurando Aplicaciones

Estructurando Aplicaciones

- En el directorio scripts teníamos un fichero para las directivas, otro para los controladores, otro para los servicios y otro para los filtros

Estructurando Aplicaciones

- Eso está bien cuando la aplicación es pequeña, sencilla y no ha crecido lo suficiente como para tener que empezar estructurar de manera seria nuestro código

Estructurando Aplicaciones

- Diseñamos la arquitectura de nuestra aplicación pensando en componentes y en features

Estructurando Aplicaciones

- Google comparte un documento donde expone su recomendación a la hora de estructurar aplicaciones Angular

<https://docs.google.com/document/d/1XXMvReO8-Awi1EZXAXS4PzDzdNvV6pGcuaF4Q9821Es/pub>

Estructurando Aplicaciones

- Estructura jerárquica recursiva formada por 2 tipos de directorios
 - Directorio "componentes"
 - O directorio "sub-secciones"

Estructurando Aplicaciones

- Directorio "componentes"
 - Para los elementos comunes reutilizados en otras partes de la aplicación

Estructurando Aplicaciones

- Directorio "sub-secciones"
 - Nombramos de manera totalmente significativa.

Estructurando Aplicaciones

- Directorio "sub-secciones"
 - Son anidados y que representan elementos estructurales "vistas" o rutas dentro de la aplicación

Estructurando: Google

- Componentes
 - Contiene servicios, directivas y filtros
 - Contiene datos comunes (imágenes, modelos etc)

Estructurando: Google

- Componentes
 - Definición de módulos Angular

Estructurando: Google

- Subsecciones
 - Contienen solo templates (.html y .css), controladores y definiciones de módulos Angular

Estructurando: Google

- Subsecciones
 - Creamos subniveles hijos de subsecciones repitiendo la misma estructura básica de subsección

Estructurando: Google

- Subsecciones
 - Se ve de manera clara la estructura jerárquica de elementos UI

Ejercicios

- Crea una aplicación Angular que sea un contenedor de generadores de número aleatorios. Vamos a crear nuestra propia estructura. No 100% como lo define Google.

Ejercicios

- Queremos que cada App que hemos hecho hasta ahora sea un módulo
- En **/client**, crea un módulo Angular para tu App

Ejercicios

- En **/client**, crea una carpeta **/modulos** que a su vez contenga 4 directorios, uno por cada app desarrollada hasta ahora

Ejercicios

- Instálale la aplicación en Nodejs y compara la parte del cliente con tu nueva aplicación angular estructurada por módulos

Múltiples views

- Pero...y en una SPA...¿Como puedo hacer para mostrar diferentes páginas?
- ¿Y si quiero algo asi como `http://mywebsite/page-x?`

Múltiples views

- Nuestra aplicación va creciendo y siendo cada vez más compleja
- Ahora tenemos un contenedor de GNAs

Múltiples views

- Y queremos poder ejecutar cada GNA en nuestra aplicación como si accediéramos a una sección diferente, a una página nueva

Múltiples views

- El siguiente paso en la construcción de la aplicación es agregar algo que nos permita "incrustar" templates en una parte del DOM, dependiendo de la URL

Enrutando: \$route

- **\$routeProvider** se utiliza para unir URLs con controladores y vistas (HTMLs parciales)
- Está observando **\$location.url()** y mapea la URL con una definición de ruta existente

Enrutando: \$route

```
myApp.config(function($locationProvider, $routeProvider)
{
    $routeProvider
        // route for Subsection 1
        .when('/subsection1', { templateUrl:
'subsections/subsection1.html', controller:
'Subsection1Ctrl' })
})
```


Enrutando: \$route

```
// route for Subsection 2
.when('/subsection2', { templateUrl:
'subsections/subsection2.html', controller:
'Subsection2Ctrl', })
// Otherwise -> go to Subsection N
.otherwise({redirectTo: '/subsectionN'});
```

Enrutando: `$route`

- El servicio `$route` se utiliza generalmente en combinación con la directriz `ngView`.

Enrutando: \$route

- El papel de la directiva ngView es incluir la plantilla de la vista de la ruta actual en su template

Enrutando: \$route

- ngRoute se define en su propio módulo y debe ser instanciado a parte: **angular-route.js**

Enrutando: \$route

```
<div class="container-fluid">  
  <div ng-view></div>  
</div>
```

Ejercicios

- Añade el router que le falta a la aplicación, para poder cargar los diferentes módulos en el contenido central de la aplicación cuando la url corresponda a cada uno de ellos

Ejercicios

- Añade `$location.path(moduleURL)` al metodo `goToModule(module)` en la directiva `topnavbar` para que este actualice la URL con la URL correspondiente al módulo y el router pueda escuchar dicho cambio y enrutar