Instituto Nacional de Aprendizaje

Curso: Programador de aplicaciones informáticas

Módulo: Programación Orientada a Objetos

Tema Angular

Profesor: Luis Alonso Bogantes Rodríguez

Alumno: Claudio Mauricio Jiménez Castro

2023

Contenido

Introducción 1
Componentes 2
Módulos 3
Directivas 4
Servicios 4
Enrutamiento 5
Formularios 6
Pipes 6
Interacción con servidores 7
Pruebas y depuración 8
Conclusiones 10
Webgrafía 11

Introducción

Angular es un framework de JavaScript de código abierto utilizado para crear aplicaciones web de una sola página (Single Page Applications, SPA). Fue desarrollado por Google y lanzado por primera vez en 2010 bajo el nombre de AngularJS. Posteriormente, en 2016, se lanzó una versión completamente reescrita y mejorada llamada simplemente Angular.

Angular permite a los desarrolladores crear aplicaciones web complejas y dinámicas de manera eficiente, gracias a su arquitectura basada en componentes y su enfoque en la separación de responsabilidades (Separation of Concerns, SoC). Los componentes en Angular son piezas reutilizables de código que contienen tanto la lógica como la interfaz de usuario de una parte específica de una aplicación.

Angular también cuenta con un sistema de enlace de datos bidireccional que permite que los cambios realizados en la interfaz de usuario se reflejen automáticamente en la lógica de la aplicación y viceversa. Además, incluye características como la inyección de dependencias, el manejo de formularios, las pruebas unitarias y la integración con otras bibliotecas y frameworks.

Una de las mayores ventajas de Angular es su gran comunidad de desarrolladores y su documentación completa y detallada, lo que hace que sea fácil encontrar soluciones a cualquier problema que surja durante el desarrollo. Además, gracias a su popularidad, hay una gran cantidad de recursos y herramientas disponibles para Angular, lo que facilita aún más su aprendizaje y uso.

Para abarcar los puntos más relevantes sobre Angular hemos de referirnos a varios subtemas de gran importancia, dentro de cuales tenemos: componentes, módulos, Directivas, servicios, enrutamiento, formularios, Pipes, interacción con servidores, pruebas y depuración.

A continuación, se detallan los aspectos más relevantes sobre estos.

Componentes

Definición y estructura de un componente: Un componente en Angular es una unidad lógica y visual que encapsula la funcionalidad y la vista de una parte de la interfaz de usuario de la aplicación. Se define mediante una clase que incluye propiedades y métodos para manipular los datos y una plantilla que define la vista que se mostrará al usuario. Además, los componentes pueden tener estilos asociados y metadatos que definen su comportamiento.

Ciclo de vida de un componente: El ciclo de vida de un componente en Angular es el conjunto de eventos que ocurren desde su creación hasta su destrucción. Los principales eventos son: ngOnChanges, ngOnInit, ngDoCheck, ngAfterContentInit, ngAfterContentChecked, ngAfterViewInit, ngAfterViewChecked, ngOnDestroy. Cada uno de ellos corresponde a un momento específico del ciclo de vida del componente y permite realizar acciones específicas en ese momento.

Uso de directivas en componentes: Las directivas son instrucciones que se aplican a elementos HTML para modificar su comportamiento o su apariencia. En Angular, existen dos tipos de directivas: las directivas estructurales y las directivas de atributo. Las directivas estructurales modifican la estructura del DOM, mientras que las directivas de atributo modifican el comportamiento o la apariencia de los elementos.

Comunicación entre componentes: Los componentes pueden comunicarse entre sí de varias maneras en Angular. Una de las formas más comunes es mediante la comunicación de eventos, en la que un componente emite un evento y otro componente lo recibe y realiza una acción. También es posible comunicar componentes mediante servicios, que actúan como intermediarios entre los componentes y permiten compartir datos y funcionalidades entre ellos.

Creación y uso de servicios en componentes: Los servicios son clases que proveen funcionalidades que pueden ser utilizadas por varios componentes de la aplicación. En Angular, los servicios se definen mediante la inyección de dependencias y se pueden utilizar en los componentes mediante la declaración de una variable que referencia al servicio. Los servicios pueden proporcionar datos, lógica de negocio, comunicación con servicios externos, entre otros

Módulos

Los módulos son una parte fundamental en la arquitectura de Angular, ya que permiten organizar y encapsular diferentes funcionalidades de la aplicación en módulos separados. Un módulo se define como una colección de componentes, directivas, servicios y otros módulos relacionados, que trabajan juntos para proporcionar una funcionalidad específica.

La estructura básica de un módulo incluye un archivo TypeScript que define el módulo y sus dependencias, y un archivo HTML que representa el template del módulo. Además, un módulo puede tener un archivo CSS para dar estilo al template y archivos adicionales que se necesiten para la funcionalidad del módulo.

En Angular, se pueden crear módulos personalizados para encapsular diferentes partes de la aplicación. Estos módulos se pueden importar en otros módulos para utilizar su funcionalidad. La importación y exportación de módulos se realiza mediante la función import de TypeScript, y puede incluirse también en el archivo NgModule del módulo.

Es importante organizar los módulos en una jerarquía adecuada, para que sea fácil mantener y escalar la aplicación. Los módulos se pueden agrupar en diferentes niveles, dependiendo de su función y su relación con otros módulos.

La inyección de dependencias es otra característica importante de los módulos en Angular. Permite que un componente o servicio tenga acceso a otros componentes o servicios que son necesarios para su funcionamiento. La inyección de dependencias se realiza mediante la función de TypeScript, que proporciona un mecanismo para crear instancias de servicios y componentes y utilizarlos en otros módulos o componentes.

Se podría aseverar entonces que los módulos en Angular permiten organizar y encapsular diferentes partes de la aplicación, facilitan la importación y exportación de funcionalidades y proporcionan un mecanismo de inyección de dependencias para compartir servicios y componentes entre diferentes partes de la aplicación.

Directivas

Las directivas son una parte fundamental de Angular que permiten agregar comportamiento a los elementos del DOM. En Angular, hay dos tipos de directivas: directivas estructurales y directivas atributivas.

Las directivas estructurales permiten modificar la estructura del DOM, agregando, eliminando o reemplazando elementos. Ejemplos de directivas estructurales en Angular son *ngIf, *ngFor y *ngSwitch.

Las directivas atributivas, por otro lado, se utilizan para agregar comportamiento a los elementos existentes en el DOM. Ejemplos de directivas atributivas son ngModel, ngStyle y ngClass. Además, es posible crear directivas personalizadas en Angular. Estas directivas se crean utilizando la API de Directivas de Angular y se pueden utilizar en la misma forma que las directivas integradas de Angular.

La comunicación entre directivas y componentes se realiza mediante la API de Input y Output de Angular. Los componentes pueden pasar datos a las directivas utilizando la propiedad Input, mientras que las directivas pueden enviar datos a los componentes utilizando la propiedad Output.

Angular también cuenta con directivas integradas que se pueden utilizar para agregar comportamiento a los elementos del DOM. Algunos ejemplos de directivas integradas de Angular son ngIf, ngFor, ngClass y ngStyle.

Servicios

Los servicios son uno de los pilares fundamentales de Angular. Son clases que se encargan de proveer datos y lógica de negocio a diferentes componentes de la aplicación.

La estructura de un servicio es similar a la de un componente, con la diferencia de que los servicios no tienen una vista asociada. En cambio, su función principal es la de encapsular una funcionalidad específica y proporcionarla a los componentes que la necesiten.

En Angular, los servicios se crean y se utilizan a través de la inyección de dependencias, lo que permite una gestión eficiente de los recursos y una mayor escalabilidad. Además, se pueden crear servicios personalizados para adaptarse a las necesidades de la aplicación.

Los servicios también permiten la comunicación entre componentes y directivas, ya que pueden actuar como intermediarios para compartir datos y funcionalidades. Por último, Angular proporciona una serie de servicios integrados, como el servicio de HTTP para realizar peticiones a servidores y el servicio de enrutamiento para la navegación entre vistas de la aplicación.

Enrutamiento

Enrutamiento en Angular es una técnica que permite la navegación entre componentes de manera organizada y jerárquica. Consiste en definir rutas que representen los diferentes componentes de una aplicación Angular, y luego permitir que el usuario navegue a través de ellas haciendo clic en los enlaces.

La estructura del enrutamiento en Angular se basa en un conjunto de clases y componentes que se encargan de manejar las rutas. El archivo de configuración de rutas es donde se definen las diferentes rutas y componentes a los que se accede a través de ellas.

Entre los conceptos más importantes del enrutamiento en Angular se encuentran la definición de rutas y navegación entre componentes. Además, el enrutamiento también permite el uso de parámetros y guardas para controlar el acceso a determinadas rutas y componentes.

Otra característica interesante del enrutamiento en Angular es la posibilidad de crear enrutamientos anidados y realizar la carga perezosa de componentes, lo que permite una mejor organización y modularidad de la aplicación.

Finalmente, el enrutamiento es una parte esencial de cualquier aplicación Angular completa, ya que permite una navegación clara y organizada entre los diferentes componentes y funcionalidades de la aplicación

Formularios

Los formularios son una parte importante en cualquier aplicación web y Angular no es la excepción. En Angular, los formularios se basan en el enlace de datos bidireccional, lo que significa que la información se puede actualizar en ambos sentidos, desde la vista a los datos y desde los datos a la vista.

En Angular, los formularios se construyen a través de directivas de formularios, que proporcionan una interfaz para interactuar con los datos del formulario. Algunas de las directivas más importantes son ngForm, ngModel y ngSubmit.

La validación de formularios es una característica importante en cualquier aplicación y Angular proporciona una forma fácil de validar formularios. Se pueden utilizar directivas de validación incorporadas o se pueden crear validaciones personalizadas.

La comunicación entre componentes y formularios es una tarea común en cualquier aplicación. En Angular, esto se puede lograr a través de la utilización de servicios, que permiten a los componentes interactuar con los datos del formulario y realizar tareas como la validación o el envío de datos.

En Angular, también es posible crear formularios personalizados, lo que permite a los desarrolladores crear soluciones de formularios que se adapten a las necesidades específicas de la aplicación. Con las herramientas de enlace de datos y las directivas de formularios personalizadas, Angular proporciona un conjunto completo de herramientas para manejar los formularios en una aplicación web.

Pipes

En Angular, los pipes son una herramienta útil para transformar datos antes de mostrarlos en la interfaz de usuario. Un pipe recibe un valor de entrada y lo transforma en otro valor de salida. Los pipes pueden ser integrados por Angular o personalizados para satisfacer las necesidades específicas de una aplicación.

Existen varios tipos de pipes en Angular, como el pipe Date, el pipe UpperCase, el pipe LowerCase, el pipe Currency, entre otros. Además, es posible crear pipes personalizados, para esto se deben implementar la interfaz PipeTransform.

Los pipes integrados en Angular pueden ser utilizados en la aplicación sin necesidad de crearlos, simplemente importando el módulo correspondiente en el componente que se esté desarrollando. Por otro lado, los pipes personalizados deben ser declarados en el módulo correspondiente y luego pueden ser utilizados en cualquier componente de la aplicación.

Es importante tener en cuenta que los pipes no deben ser utilizados para lógica compleja, ya que esto puede afectar el rendimiento de la aplicación. En su lugar, deben ser utilizados para tareas simples como el formateo de fechas o números.

En resumen, los pipes en Angular son una herramienta útil para transformar datos antes de mostrarlos en la interfaz de usuario. Los pipes pueden ser integrados por Angular o personalizados, y deben ser utilizados para tareas simples de formateo de datos.

Interacción con servidores

Angular ofrece la posibilidad de interactuar con servidores para realizar peticiones HTTP y recibir respuestas. Para ello, se utilizan los módulos HttpClient y HttpInterceptor.

El módulo HttpClient proporciona una forma sencilla de realizar solicitudes HTTP, tanto para enviar como para recibir datos. Además, es posible configurar el comportamiento de las peticiones mediante opciones como headers, parámetros de query, cuerpo de la petición, entre otros.

El módulo HttpInterceptor permite interceptar las solicitudes HTTP antes de ser enviadas al servidor y también las respuestas antes de ser entregadas al cliente. Esto puede ser útil para realizar tareas como agregar encabezados de autorización o manejar errores de forma centralizada.

Para trabajar con servicios REST en Angular, se pueden utilizar los métodos proporcionados por el módulo HttpClient, como get(), post(), put(), delete(), entre otros. También es posible configurar servicios REST para que trabajen con diferentes tipos de datos, como JSON o XML.

El manejo de errores y excepciones es importante en cualquier aplicación, y en Angular se puede hacer mediante la implementación de un interceptor para manejar los errores HTTP y devolver una respuesta personalizada al usuario.

Finalmente, es posible integrar una aplicación Angular con un servidor externo mediante la configuración adecuada del servicio HttpClient y el envío y recepción de datos a través de HTTP.

Pruebas y depuración

Las pruebas y la depuración son una parte fundamental en el proceso de desarrollo de cualquier aplicación, y Angular no es la excepción. En este tema se abordan los diferentes tipos de pruebas que se pueden realizar en una aplicación Angular, desde las pruebas unitarias hasta las pruebas end-to-end y de aceptación.

Se explora cómo configurar un entorno de pruebas en Angular y se presentan las diferentes herramientas de depuración disponibles. Además, se brinda información sobre la realización de pruebas unitarias y de integración, y cómo estas pruebas pueden ser automatizadas para una mayor eficiencia en el proceso de desarrollo.

Asimismo, se profundiza en las pruebas end-to-end y de aceptación, que permiten simular el comportamiento de un usuario en una aplicación, y cómo estas pruebas pueden ayudar a identificar problemas en la interacción con el usuario y en el funcionamiento de la aplicación en general. También se discute cómo se pueden ejecutar pruebas en paralelo para ahorrar tiempo y mejorar la eficiencia del proceso de desarrollo.

En resumen, el tema de pruebas y depuración en Angular es fundamental para asegurar la calidad y la funcionalidad de la aplicación desarrollada.

Conclusiones

Angular es un framework de desarrollo web completo y muy eficiente que ofrece una amplia variedad de herramientas y funcionalidades para crear aplicaciones web de alta calidad. Su enfoque en módulos, la reutilización de componentes y la escalabilidad a través de la arquitectura de microservicios, lo convierte en una opción popular para proyectos de cualquier tamaño y complejidad.

La estructura jerárquica de los componentes, módulos y servicios de Angular permite una fácil organización y mantenimiento de código, mientras que el enrutamiento, las directivas y los pipes brindan una gran flexibilidad en la presentación de datos y la interacción con el usuario.

Además, Angular se integra fácilmente con servidores externos a través de su módulo HttpClient y HttpInterceptor, lo que permite una interacción sin problemas y de servicios web.

La capacidad de realizar pruebas unitarias, de integración, end-to-end y de aceptación también es un punto fuerte de Angular, ya que permite a los desarrolladores detectar y solucionar errores de manera temprana, lo que a su vez conduce a una aplicación más confiable y escalable.

En resumen, Angular es una herramienta poderosa para el desarrollo de aplicaciones web modernas y escalables, y su amplia comunidad de desarrolladores y documentación sólida lo hacen una excelente opción para cualquier proyecto web de cualquier tamaño y complejidad.

Webgrafía

Recuperado de: https://lemus.webs.upv.es/wordpress/wp-content/uploads/2020/07/angular_compressed.pdf

Recuperado de: https://tutorialesenpdf.com/angular/