Desarrollo avanzado de aplicaciones II

Tema Nº13:Módulos.

Indicador de logro Nº13:

Estructura el proyecto web mediante una arquitectura modular basado en buenas prácticas.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº13:**

**Módulos.**

**Subtema 13.1:**

Arquitectura modular.

**Ejemplos:**

Las aplicaciones de Angular son modulares y Angular tiene su propio sistema de modularidad llamado NgModules. Los NgModules son contenedores para un bloque cohesivo de código dedicado a un dominio de aplicación, un flujo de trabajo o un conjunto de capacidades estrechamente relacionadas. Pueden contener componentes, proveedores de servicios y otros archivos de código cuyo alcance está definido por el NgModule que los contiene. Pueden importar la funcionalidad que se exporta desde otros NgModules y exportar la funcionalidad seleccionada para que la utilicen otros NgModules.

Cada aplicación Angular tiene al menos una clase NgModule, el módulo raíz, que se denomina convencionalmente AppModule y reside en un archivo llamado app.module.ts. Inicie su aplicación arrancando el NgModule raíz.

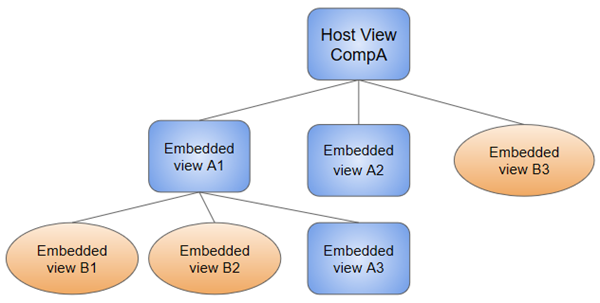
Si bien una aplicación pequeña puede tener solo un NgModule, la mayoría de las aplicaciones tienen muchos más módulos de funciones. El NgModule raíz de una aplicación se llama así porque puede incluir NgModules secundarios en una jerarquía de cualquier profundidad.

**NgModules y componentes**

NgModules proporciona un contexto de compilación para sus componentes. Un NgModule raíz siempre tiene un componente raíz que se crea durante el arranque, pero cualquier NgModule puede incluir cualquier número de componentes adicionales, que se pueden cargar a través del enrutador o crear a través de la plantilla. Los componentes que pertenecen a un NgModule comparten un contexto de compilación.



Un componente y su plantilla juntos definen una vista. Un componente puede contener una jerarquía de vistas, lo que le permite definir áreas arbitrariamente complejas de la pantalla que se pueden crear, modificar y destruir como una unidad. Una jerarquía de vistas puede mezclar vistas definidas en componentes que pertenecen a diferentes NgModules. Este suele ser el caso, especialmente para las bibliotecas de interfaz de usuario.



Cuando crea un componente, se asocia directamente con una sola vista, denominada vista de host. La vista de host puede ser la raíz de una jerarquía de vistas, que puede contener vistas incrustadas, que a su vez son las vistas de host de otros componentes. Esos componentes pueden estar en el mismo NgModule o pueden importarse desde otros NgModules. Las vistas en el árbol se pueden anidar a cualquier profundidad.

**Subtema 13.2:**

Crear un Shared Module y Core Module.

**Ejemplos:**

NgModules configuran el inyector y el compilador y ayudan a organizar las cosas relacionadas juntas.

Un NgModule es una clase marcada por el decorador @NgModule. @NgModule toma un objeto de metadatos que describe cómo compilar la plantilla de un componente y cómo crear un inyector en tiempo de ejecución. Identifica los propios componentes, directivas y canalizaciones del módulo, haciendo públicos algunos de ellos, a través de la propiedad de exportaciones, para que los componentes externos puedan utilizarlos. @NgModule también puede agregar proveedores de servicios a los inyectores de dependencia de la aplicación.

Para ver una aplicación de ejemplo que muestra todas las técnicas que cubren las páginas relacionadas con NgModules, vea el ejemplo en vivo / ejemplo de descarga. Para obtener explicaciones sobre las técnicas individuales, visite las páginas correspondientes de NgModule en la sección NgModules.

Modularidad angular

Los módulos son una excelente manera de organizar una aplicación y extenderla con capacidades de bibliotecas externas.

Las bibliotecas angulares son NgModules, como FormsModule, HttpClientModule y RouterModule. Muchas bibliotecas de terceros están disponibles como NgModules, como Material Design, Ionic y AngularFire2.

Los NgModules consolidan componentes, directivas y conductos en bloques cohesivos de funcionalidad, cada uno de los cuales se centra en un área de funciones, un dominio de negocio de aplicaciones, un flujo de trabajo o una colección común de utilidades.

Los módulos también pueden agregar servicios a la aplicación. Dichos servicios pueden desarrollarse internamente, como algo que desarrollaría usted mismo o que provenga de fuentes externas, como el enrutador angular y el cliente HTTP.

Los módulos se pueden cargar con entusiasmo cuando se inicia la aplicación o el enrutador puede cargarlos de forma diferida de forma asíncrona.

Los metadatos de NgModule hacen lo siguiente:

* Declara qué componentes, directivas y canalizaciones pertenecen al módulo.
* Hace que algunos de esos componentes, directivas y canalizaciones sean públicos para que las plantillas de componentes de otros módulos puedan usarlos.
* Importa otros módulos con los componentes, directivas y conductos que necesitan los componentes del módulo actual.
* Proporciona servicios que pueden utilizar otros componentes de la aplicación.

Cada aplicación Angular tiene al menos un módulo, el módulo raíz. Arranca ese módulo para iniciar la aplicación.

La CLI de Angular genera el siguiente módulo de aplicación básico al crear una nueva aplicación.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

En la parte superior están las declaraciones de importación. La siguiente sección es donde configura el @NgModule indicando qué componentes y directivas le pertenecen (declaraciones), así como qué otros módulos usa (importaciones). Para obtener más información sobre la estructura de un @NgModule, asegúrese de leer Bootstrapping.

**Subtema 13.3:**

Feature Modules Routing.

**Ejemplos:**

A medida que su aplicación crece, puede organizar el código relevante para una función específica. Esto ayuda a aplicar límites claros para las funciones. Con los módulos de funciones, puede mantener el código relacionado con una función o función específica separada de otro código. La delimitación de áreas de su aplicación ayuda a la colaboración entre desarrolladores y equipos, separando directivas y administrando el tamaño del módulo raíz.

Módulos de funciones frente a módulos raíz

Un módulo de funciones es una mejor práctica organizacional, a diferencia de un concepto de la API angular principal. Un módulo de funciones ofrece un conjunto coherente de funciones centradas en una necesidad de aplicación específica, como un flujo de trabajo de usuario, enrutamiento o formularios. Si bien puede hacer todo dentro del módulo raíz, los módulos de funciones lo ayudan a dividir la aplicación en áreas específicas. Un módulo de funciones colabora con el módulo raíz y con otros módulos a través de los servicios que proporciona y los componentes, directivas y canalizaciones que comparte.

Cómo hacer un módulo de funciones

Suponiendo que ya tiene una aplicación que creó con la CLI de Angular, cree un módulo de funciones usando la CLI ingresando el siguiente comando en el directorio raíz del proyecto. Reemplace CustomerDashboard con el nombre de su módulo. Puede omitir el sufijo "Módulo" del nombre porque la CLI lo agrega:

Imagen que contiene Logotipo

Descripción generada automáticamente

Esto hace que la CLI cree una carpeta llamada customer-dashboard con un archivo dentro llamado customer-dashboard.module.ts con el siguiente contenido:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

La estructura de un NgModule es la misma, ya sea un módulo raíz o un módulo de funciones. En el módulo de funciones generado por CLI, hay dos declaraciones de importación de JavaScript en la parte superior del archivo: la primera importa NgModule, que, al igual que el módulo raíz, le permite usar el decorador @NgModule; el segundo importa CommonModule, que aporta muchas directivas comunes como ngIf y ngFor. Los módulos de funciones importan CommonModule en lugar de BrowserModule, que solo se importa una vez en el módulo raíz. CommonModule solo contiene información para directivas comunes como ngIf y ngFor, que se necesitan en la mayoría de las plantillas, mientras que BrowserModule configura la aplicación Angular para el navegador, que debe hacerse solo una vez.

La matriz de declaraciones está disponible para que agregue declarables, que son componentes, directivas y canalizaciones que pertenecen exclusivamente a este módulo en particular. Para agregar un componente, ingrese el siguiente comando en la línea de comandos donde el panel de cliente es el directorio donde la CLI generó el módulo de funciones y el Panel de cliente es el nombre del componente:

Imagen que contiene Texto

Descripción generada automáticamente

Esto genera una carpeta para el nuevo componente dentro de la carpeta del panel del cliente y actualiza el módulo de funciones con la información CustomerDashboardComponent:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

El componente del panel de control del cliente ahora está en la lista de importación de JavaScript en la parte superior y se agregó a la matriz de declaraciones, lo que le permite a Angular asociar este nuevo componente con este módulo de funciones.

Para incorporar el módulo de funciones en su aplicación, debe informar al módulo raíz, app.module.ts. Observe la exportación CustomerDashboardModule en la parte inferior de customer-dashboard.module.ts. Esto lo expone para que otros módulos puedan acceder a él. Para importarlo al AppModule, agréguelo a las importaciones en app.module.ts y a la matriz de importaciones:

Imagen que contiene Texto

Descripción generada automáticamente

**Subtema 13.4:**

Lazy-loading feature modules.

**Ejemplos:**

En la aplicación básica generada por CLI, los módulos se cargan con entusiasmo, lo que significa que todos se cargan cuando se inicia la aplicación. Angular usa un sistema de inyectores para hacer que las cosas estén disponibles entre módulos. En una aplicación cargada con entusiasmo, el inyector de la aplicación raíz hace que todos los proveedores en todos los módulos estén disponibles en toda la aplicación.

Este comportamiento cambia necesariamente cuando usa la carga diferida. La carga diferida es cuando carga módulos solo cuando los necesita; por ejemplo, al enrutar. No se cargan de inmediato, como ocurre con los módulos cargados con entusiasmo. Esto significa que los servicios enumerados en sus matrices de proveedores no están disponibles porque el inyector raíz no conoce estos módulos.

Cuando el enrutador angular carga un módulo de forma diferida, crea un nuevo inyector. Este inyector es un hijo del inyector de aplicación raíz. Imagínese un árbol de inyectores; hay un solo inyector de raíz y luego un inyector secundario para cada módulo de carga diferida. El enrutador agrega todos los proveedores desde el inyector raíz hasta el inyector secundario. Cuando el enrutador crea un componente dentro del contexto de carga diferida, Angular prefiere las instancias de servicio creadas a partir de estos proveedores a las instancias de servicio del inyector raíz de la aplicación.

Cualquier componente creado dentro del contexto de un módulo de carga diferida, como la navegación del enrutador, obtiene la instancia local del servicio, no la instancia en el inyector de la aplicación raíz. Los componentes de los módulos externos continúan recibiendo la instancia creada para la raíz de la aplicación.

Aunque puede proporcionar servicios mediante módulos de carga diferida, no todos los servicios pueden cargarse de forma diferida. Por ejemplo, algunos módulos solo funcionan en el módulo raíz, como el enrutador. El enrutador trabaja con el objeto de ubicación global en el navegador.

Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

**Subtema 13.5:**

Preloading modules.

**Ejemplos:**

La precarga mejora la UX al cargar partes de su aplicación en segundo plano. Puede precargar módulos o datos de componentes.

La precarga de módulos mejora la UX al cargar partes de su aplicación en segundo plano para que los usuarios no tengan que esperar a que se descarguen los elementos cuando activan una ruta.

Para habilitar la precarga de todos los módulos cargados de forma diferida, importe el token PreloadAllModules desde el enrutador angular.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Aún en AppRoutingModule, especifique su estrategia de precarga en forRoot ().

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Precarga de datos de componentes

Para precargar datos de componentes, puede utilizar un solucionador. Los resolutores mejoran la experiencia de usuario al bloquear la carga de la página hasta que todos los datos necesarios estén disponibles para mostrar la página por completo.

Resolvers

Crea un servicio de resolución. Con la CLI, el comando para generar un servicio es el siguiente:

Texto

Descripción generada automáticamente con confianza baja

En el servicio recién creado, implemente la interfaz Resolve proporcionada por el paquete @ angular / router:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Importe este resolutor al módulo de enrutamiento de su módulo.



Agregue un objeto de resolución a la configuración de ruta del componente.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En el constructor del componente, inyecte una instancia de la clase ActivatedRoute que represente la ruta actual.

Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente

Utilice la instancia inyectada de la clase ActivatedRoute para acceder a los datos asociados con una ruta determinada.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Actividad:**

a) CUESTIONARIO TÉCNICO

* Explique con sus propias palabras una Arquitectura Modular.
* Explique Lazy-loading.
* Explique Preloading Module.
* Cree un nuevo proyecto en Angular y añada un SharedModule.
* Cree un nuevo proyecto en Angular y añada un ModuleRouting.

b) CONCLUSIONES DE LA EXPERIENCIA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_