Angular.io

resumen de arquitectura

Contenido

Angular es una plataforma y un marco para crear aplicaciones cliente en HTML y TypeScript. Angular está escrito en TypeScript. Implementa la funcionalidad básica y opcional como un conjunto de bibliotecas TypeScript que importa a sus aplicaciones.

Los componentes básicos de una aplicación angular son *NgModules* , que proporcionan un contexto de compilación para *componentes* .

NgModules recopila código relacionado en conjuntos funcionales;

Una aplicación angular se define mediante un conjunto de NgModules.

Una aplicación siempre tiene al menos un *módulo raíz* que permite el arranque, y generalmente tiene muchos más *módulos de características* .

- **Los componentes definen *vistas* , que son conjuntos de elementos de pantalla que Angular puede elegir y modificar según la lógica y los datos de su programa.**
- **Los componentes utilizan *servicios* , que proporcionan una funcionalidad específica que no está directamente relacionada con las vistas. Los proveedores de servicios pueden *inyectarse* en los**

componentes como *dependencias* , haciendo que su código sea modular, reutilizable y eficiente.

Tanto los componentes como los servicios son simplemente clases, con *decoradores* que marcan su tipo y proporcionan metadatos que le indican a Angular cómo usarlos.

- **Los metadatos para una clase de componente lo asocian con una *plantilla* que define una vista. Una plantilla combina HTML ordinario con *directivas* angulares y *marcas de enlace* que permiten a Angular**

modificar el HTML antes de presentarlo para su visualización.

- **Los metadatos para una clase de servicio proporcionan la información que Angular necesita para ponerla a disposición de los componentes a través de *la inyección de dependencia (DI)* .**

componentes de una aplicación suelen definir muchas vistas, ordenadas jerárquicamente. Angular proporciona el *Router* servicio para ayudarlo a definir rutas de navegación entre vistas. El enrutador proporciona capacidades de navegación sofisticadas en el navegador.

módulos

Los *NgModules* angulares difieren y complementan los módulos JavaScript (ES2015). Un NgModule declara un contexto de compilación para un conjunto de componentes que está dedicado a un dominio de aplicación, un flujo de trabajo o un conjunto de capacidades estrechamente relacionado. Un NgModule puede asociar sus componentes con código relacionado, como servicios, para formar unidades funcionales.

Cada aplicación Angular tiene un *módulo raíz* , (root module), denominado convencionalmente AppModule, que proporciona el mecanismo de arranque que inicia la aplicación. Una aplicación generalmente contiene muchos módulos funcionales.

Al igual que los módulos de JavaScript, NgModules puede importar la funcionalidad de otros NgModules y permitir que su propia funcionalidad sea exportada y utilizada por otros NgModules.

Por ejemplo, para usar el servicio de enrutador en su aplicación, importa el Router NgModule.

Organizar su código en distintos módulos funcionales ayuda a gestionar el desarrollo de aplicaciones complejas y a diseñar para su reutilización. Además, esta técnica le permite aprovechar *la carga diferida* , es decir, cargar módulos a pedido, para minimizar la cantidad de código que debe cargarse al inicio.

Para una discusión más detallada, vea Introducción a los módulos .

componentes

Cada aplicación angular tiene al menos un componente, el *componente raíz* que conecta una jerarquía de componentes con el modelo de objeto de documento de página (DOM). Cada componente define una clase que contiene datos y lógica de la aplicación, y está asociada con una *plantilla* HTML que define una vista que se mostrará en un entorno de destino.

El decorador @Component()

identifica la clase inmediatamente debajo de ella como un componente y proporciona la plantilla y los metadatos específicos de los componentes relacionados

Los decoradores son funciones que modifican las clases de JavaScript. Angular define una serie de decoradores que adjuntan tipos específicos de metadatos a las clases, para que el sistema sepa qué significan esas clases y cómo deberían funcionar.

Templates, directives y data binding

Una plantilla (template) combina HTML con marcado angular que puede modificar elementos HTML antes de que se muestren.

Las *directivas de* plantilla proporcionan lógica de programa, y el *marcado vinculante* (data binding) conecta los datos de su aplicación y el DOM.

Hay dos tipos de enlace de datos (data binding):

- ***El enlace de eventos*** **(event binding) permite que su aplicación responda a la entrada del usuario en el entorno objetivo actualizando los datos de su aplicación.**

- ***El enlace de propiedad (Property binding) le*** **permite interpolar valores que se calculan a partir de los datos de su aplicación en el HTML.**

Antes de que se muestre una vista, Angular evalúa las directivas y resuelve la sintaxis de enlace en la plantilla para modificar los elementos HTML y el DOM, de acuerdo con los datos y la lógica de su programa. Angular admite el *enlace de datos bidireccional* , lo que significa que los cambios en el DOM, como las elecciones del usuario, también se reflejan en los datos de su programa.

Sus plantillas pueden usar *tuberías* para mejorar la experiencia del usuario mediante la transformación de valores para su visualización. Por ejemplo, use canalizaciones para mostrar fechas y valores de moneda que sean apropiados

para la configuración regional de un usuario. Angular proporciona tuberías predefinidas para transformaciones comunes, y también puede definir sus propias tuberías.

Para una discusión más detallada de estos conceptos, vea Introducción a los componentes .

Servicios y inyección de dependencia

Para los datos o la lógica que no están asociados con una vista específica y que desea compartir entre componentes, cree una clase de *servicio* . Una definición de clase de servicio es precedida inmediatamente por el decorador. El decorador proporciona los metadatos que permiten que otros proveedores se inyecten como dependencias en su clase

@Injectable()

La inyección de dependencia **(DI) le permite mantener sus clases de componentes esbeltas y eficientes. No obtienen datos del servidor, validan la entrada del usuario o inician sesión directamente en la consola; delegan tales tareas a los servicios.**

Para una discusión más detallada, vea Introducción a los servicios y DI .

Enrutamiento (Routing)

Angular Router NgModule proporciona un servicio que le permite definir una ruta de navegación entre los diferentes estados de la aplicación y ver jerarquías en su aplicación. Se basa en las convenciones de navegación del navegador familiares:

- **Ingrese una URL en la barra de direcciones y el navegador navega a la página correspondiente.**
- **Haga clic en los enlaces de la página y el navegador navega a una nueva página.**

- **Haga clic en los botones de retroceso y avance del navegador y el navegador navega hacia atrás y hacia adelante a través del historial de páginas que ha visto.**

El enrutador asigna rutas (paths) similares a URL a vistas en lugar de páginas. Cuando un usuario realiza una acción, como hacer clic en un enlace, que cargaría una nueva página en el navegador, el enrutador intercepta el comportamiento del navegador y muestra u oculta las jerarquías de vista.

Si el enrutador determina que el estado actual de la aplicación requiere una funcionalidad particular, y el módulo que lo define no se ha cargado, el enrutador puede *cargar* el módulo bajo demanda según la demanda.

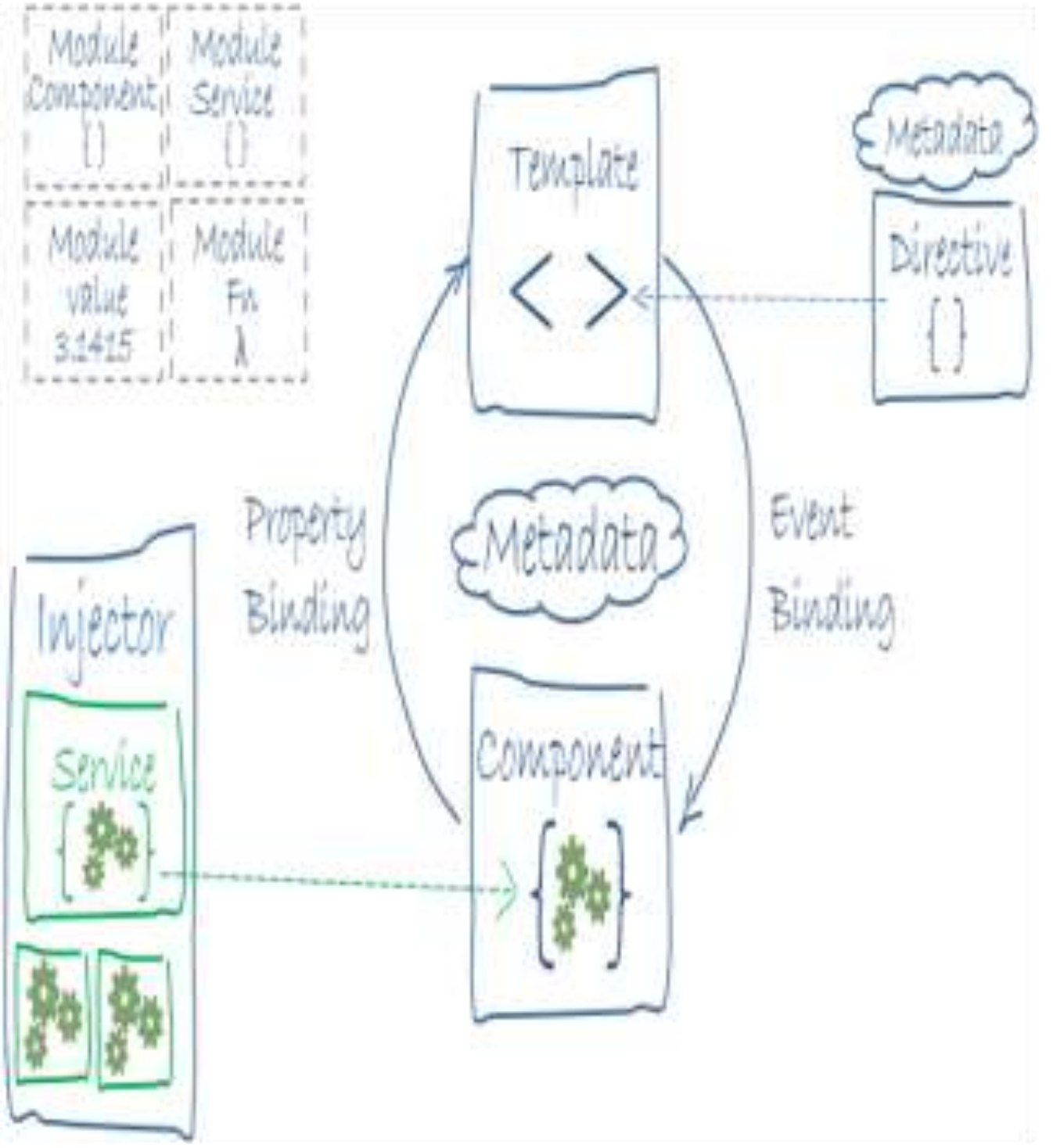
El enrutador interpreta una URL de enlace de acuerdo con las reglas de navegación de visualización de la aplicación y el estado de los datos. Puede navegar a nuevas vistas cuando el usuario hace clic en un botón o selecciona desde un cuadro desplegable, o en respuesta a algún otro estímulo de cualquier fuente.

El enrutador registra la actividad en el historial del navegador, por lo que los botones de retroceso y avance también funcionan.

Para definir reglas de navegación, asocie *rutas de navegación* con sus componentes. Una ruta utiliza una sintaxis similar a una URL que integra los datos de su programa, de la misma manera que la sintaxis de la plantilla integra sus vistas con los datos de su programa. Luego puede aplicar la lógica del programa para elegir qué vistas mostrar u ocultar, en respuesta a la entrada del usuario y sus propias reglas de acceso.

¿Cuál es el siguiente

Has aprendido los conceptos básicos sobre los principales bloques de construcción de una aplicación angular. El siguiente diagrama muestra cómo se relacionan estas piezas básicas.



***Juntos, un componente y una plantilla definen una vista angular.**

***Un decorador en una clase de componente agrega los metadatos, incluido un puntero a la plantilla asociada.**

- **Las directivas y el marcado vinculante en la plantilla de un componente modifican las vistas en función de los datos y la lógica del programa.**

- **El inyector de dependencia proporciona servicios a un componente, como el servicio de enrutador que le permite definir la navegación entre vistas.**

Cada uno de estos temas se presenta con más detalle en las siguientes páginas.

- **Introducción a los módulos**
- **Introducción a componentes**
    - **Plantillas y vistas**
    - **Metadatos de componentes**
    - **El enlace de datos**
    - **Directivas**
    - **Tubería**
- **Introducción a los servicios e inyección de dependencias.**

Tenga en cuenta que el código al que se hace referencia en estas páginas está disponible como ejemplo en vivo / ejemplo de descarga.

Cuando esté familiarizado con estos bloques de construcción fundamentales, puede explorarlos con más detalle en la documentación. Para obtener más información sobre las herramientas y técnicas que están disponibles para ayudarlo a construir e implementar aplicaciones angulares, consulte los siguientes pasos: herramientas y técnicas