Desarrollo avanzado de aplicaciones II

Tema Nº12:Formularios y modelos.

Indicador de logro Nº12:

Define formularios en angular utilizando sus elementos y manipula los objetos a través de Typescript.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº12:**

**Formularios y modelos.**

**Subtema 12.1:**

Reactive forms vs Template-driven forms.

Manejar la entrada del usuario con formularios es la piedra angular de muchas aplicaciones comunes. Las aplicaciones usan formularios para permitir a los usuarios iniciar sesión, actualizar un perfil, ingresar información confidencial y realizar muchas otras tareas de ingreso de datos.

Angular proporciona dos enfoques diferentes para manejar la entrada del usuario a través de formularios: reactive and template-driven. Ambos capturan los eventos de entrada del usuario desde la vista, validan la entrada del usuario, crean un modelo de formulario y un modelo de datos para actualizar y proporcionan una forma de realizar un seguimiento de los cambios.

Elegir un enfoque

Los formularios reactivos y los formularios basados en plantillas procesan y administran los datos del formulario de manera diferente. Cada enfoque ofrece diferentes ventajas.

* Reactive forms, proporcionan acceso directo y explícito al modelo de objetos de formularios subyacentes. En comparación con los formularios basados en plantillas, son más robustos: son más escalables, reutilizables y comprobables. Si los formularios son una parte clave de su aplicación, o si ya está utilizando patrones reactivos para crear su aplicación, utilice formularios reactivos.
* Template-driven forms, se basan en las directivas de la plantilla para crear y manipular el modelo de objeto subyacente. Son útiles para agregar un formulario simple a una aplicación, como un formulario de suscripción a una lista de correo electrónico. Son fáciles de agregar a una aplicación, pero no escalan tan bien como las formas reactivas. Si tiene requisitos de formulario muy básicos y lógica que se pueden administrar únicamente en la plantilla, los formularios basados en plantillas podrían ser una buena opción.

Diferencias clave

La siguiente tabla resume las diferencias clave entre los formularios reactive y template-driven forms.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Escalabilidad

Si los formularios son una parte central de su aplicación, la escalabilidad es muy importante. Es fundamental poder reutilizar modelos de formularios en todos los componentes.

Los formularios reactivos son más escalables que los formularios basados en plantillas. Proporcionan acceso directo a la API de formulario subyacente y utilizan el flujo de datos síncrono entre la vista y el modelo de datos, lo que facilita la creación de formularios a gran escala. Los formularios reactivos requieren menos configuración para las pruebas, y las pruebas no requieren un conocimiento profundo de la detección de cambios para probar correctamente las actualizaciones y la validación de los formularios.

Los formularios basados en plantillas se centran en escenarios simples y no son tan reutilizables. Extraen la API de formulario subyacente y utilizan el flujo de datos asincrónico entre la vista y el modelo de datos. La abstracción de formularios basados en plantillas también afecta las pruebas. Las pruebas dependen en gran medida de la ejecución manual de detección de cambios para ejecutarse correctamente y requieren más configuración.

**Ejemplos:**

**Subtema 12.2:**

Modelos usando TypeScript.

**Ejemplos:**

Los formularios controlados por emplate se basan en directivas definidas en FormsModule.

* La directiva NgModel reconcilia los cambios de valor en el elemento de formulario adjunto con los cambios en el modelo de datos, lo que le permite responder a la entrada del usuario con validación de entrada y manejo de errores.
* La directiva NgForm crea una instancia de FormGroup de nivel superior y la vincula a un elemento <form> para rastrear el valor agregado del formulario y el estado de validación. Tan pronto como importe FormsModule, esta directiva se activa de forma predeterminada en todas las etiquetas <form>. No es necesario agregar un selector especial.
* La directiva NgModelGroup crea y vincula una instancia de FormGroup a un elemento DOM.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

El formulario destaca algunas características de diseño que facilitan su uso. Por ejemplo, los dos campos obligatorios tienen una barra verde a la izquierda para facilitar su localización. Estos campos tienen valores iniciales, por lo que el formulario es válido y el botón Enviar está habilitado.

A medida que trabaje con este formulario, aprenderá cómo incluir lógica de validación, cómo personalizar la presentación con CSS estándar y cómo manejar las condiciones de error para garantizar una entrada válida. Si el usuario elimina el nombre del héroe, por ejemplo, el formulario deja de ser válido. La aplicación detecta el estado cambiado y muestra un error de validación en un estilo llamativo. Además, el botón Enviar está desactivado y la barra "requerida" a la izquierda del control de entrada cambia de verde a rojo.

No válido, nombre obligatorio

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

El siguiente paso es vincular los controles de entrada a las propiedades Hero correspondientes con enlace de datos bidireccional, para que respondan a la entrada del usuario actualizando el modelo de datos y también respondan a los cambios programáticos en los datos actualizando la pantalla.

La directiva ngModel declarada en FormsModule le permite vincular controles en su formulario basado en plantillas a propiedades en su modelo de datos. Cuando incluye la directiva usando la sintaxis para el enlace de datos bidireccional, [(ngModel)], Angular puede rastrear el valor y la interacción del usuario del control y mantener la vista sincronizada con el modelo.

* Edite el archivo de plantilla hero-form.component.html.
* Busque la etiqueta <input> junto a la etiqueta de nombre.
* Agregue la directiva ngModel, usando la sintaxis de enlace de datos bidireccional [(ngModel)] = "...".

Texto

Descripción generada automáticamente 

Añadir el código anterior dentro de esta sección.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Imagen que contiene Texto

Descripción generada automáticamente

**Subtema 12.3:**

Agregar un Form Control básico.

**Ejemplos:**

Las formas reactivas utilizan un enfoque explícito e inmutable para gestionar el estado de una forma en un momento dado. Cada cambio en el estado del formulario devuelve un nuevo estado, que mantiene la integridad del modelo entre cambios. Los formularios reactivos se construyen alrededor de flujos observables, donde las entradas y los valores del formulario se proporcionan como flujos de valores de entrada, a los que se puede acceder de forma síncrona.

Los formularios reactivos también brindan un camino directo a las pruebas porque tiene la seguridad de que sus datos son consistentes y predecibles cuando se solicitan. Cualquier consumidor de las transmisiones tiene acceso para manipular esos datos de forma segura.

Las formas reactivas se diferencian de las formas basadas en plantillas de distintas formas. Las formas reactivas proporcionan acceso sincrónico al modelo de datos, inmutabilidad con operadores observables y seguimiento de cambios a través de flujos observables.

Los formularios basados ​​en plantillas permiten el acceso directo para modificar datos en su plantilla, pero son menos explícitos que los formularios reactivos porque se basan en directivas integradas en la plantilla, junto con datos mutables para realizar un seguimiento de los cambios de forma asincrónica. Consulte la Descripción general de formularios para obtener comparaciones detalladas entre los dos paradigmas.

Agregar un control de formulario básico

Hay tres pasos para usar los controles de formulario.

* Registre el módulo de formularios reactivos en su aplicación. Este módulo declara las directivas de forma reactiva que necesita para usar reactive forms.
* Genere una nueva instancia de FormControl y guárdela en el componente.
* Registre FormControl en la plantilla.

Luego puede mostrar el formulario agregando el componente a la plantilla.

Los siguientes ejemplos muestran cómo agregar un único control de formulario. En el ejemplo, el usuario ingresa su nombre en un campo de entrada, captura ese valor de entrada y muestra el valor actual del elemento de control de formulario.

Registrar el módulo de formularios reactivos

Para usar controles de formulario reactivo, importe ReactiveFormsModule desde el paquete @ angular / forms y agréguelo a la matriz de importaciones de su NgModule.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza baja

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Subtema 12.4:**

Usar el servicio FormBuilder para generar controles.

**Ejemplos:**

La creación manual de instancias de control de formularios puede volverse repetitiva cuando se trata de varios formularios. El servicio FormBuilder proporciona métodos convenientes para generar controles.

Siga los siguientes pasos para aprovechar este servicio.

* Importe la clase FormBuilder.
* Inyecte el servicio FormBuilder.
* Genere el contenido del formulario.

Importe la clase FormBuilder del paquete @ angular / forms.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Inyectar el servicio FormBuilder

El servicio FormBuilder es un proveedor inyectable que se proporciona con el módulo de formularios reactivos. Inyecte esta dependencia agregándola al constructor del componente.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Generar controles de formulario

El servicio FormBuilder tiene tres métodos: control (), group () y array (). Estos son métodos de fábrica para generar instancias en sus clases de componentes, incluidos controles de formulario, grupos de formulario y matrices de formularios.

Utilice el método de grupo para crear los controles profileForm.

Texto

Descripción generada automáticamente con confianza media

**Subtema 12.5:**

FormControl, FormGroup, FormArray, FormBuilder y FormControlName.

**Ejemplos:**

Tanto los formularios reactivos como los basados en plantillas se basan en las siguientes clases base.

* FormControl rastrea el valor y el estado de validación de un control de formulario individual.
* FormGroup rastrea los mismos valores y estados para una colección de controles de formulario.
* FormArray rastrea los mismos valores y estados para una matriz de controles de formulario.
* ControlValueAccessor crea un puente entre las instancias de Angular FormControl y los elementos DOM nativos.

**Actividad:**

a) CUESTIONARIO TÉCNICO

* ¿Qué es Reactive Forms?
* ¿Qué es Templete driven form?
* Describa un caso donde utilizar FormControl.
* Cree un nuevo proyecto en Angular y utilice Reactive Forms.
* Cree un nuevo proyecto en Angular y utilice template driven forms.

b) CONCLUSIONES DE LA EXPERIENCIA

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