Desarrollo avanzado de aplicaciones II

Tema Nº11:Integración a servicios web.

Indicador de logro Nº11:

Consume información de servicios web REST, considerando el envío de parámetros y los métodos HTTP.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº11:**

**Integración a servicios web**

**Subtema 11.1:**

Configuración para la comunicación con el servidor.

La mayoría de las aplicaciones de front-end necesitan comunicarse con un servidor a través del protocolo HTTP, para descargar o cargar datos y acceder a otros servicios de back-end. Angular proporciona una API HTTP de cliente para aplicaciones Angular, la clase de servicio HttpClient en @ angular / common / http.

El servicio de cliente HTTP ofrece las siguientes características principales.

* La capacidad de solicitar objetos de respuesta escritos.
* Manejo de errores simplificado.
* Funciones de testabilidad.
* Interceptación de solicitudes y respuestas.

Antes de trabajar con HttpClientModule, debe tener un conocimiento básico de lo siguiente:

* Programación TypeScript
* Uso del protocolo HTTP
* Fundamentos del diseño de aplicaciones angulares, como se describe en Conceptos angulares
* Técnicas y operadores observables.

Antes de que pueda usar HttpClient, necesita importar el Angular HttpClientModule. La mayoría de las aplicaciones lo hacen en el AppModule raíz.

**Ejemplos:**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Luego, puede inyectar el servicio HttpClient como una dependencia de una clase de aplicación, como se muestra en el siguiente ejemplo de ConfigService.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

El servicio HttpClient utiliza observables para todas las transacciones. Debe importar los símbolos de operador y observables RxJS que aparecen en los fragmentos de código de ejemplo. Estas importaciones de ConfigService son típicas.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

**Subtema 11.2:**

Solicitar datos de un servidor (GET).

**Ejemplos:**

Utilice el método HttpClient.get () para obtener datos de un servidor. El método asincrónico envía una solicitud HTTP y devuelve un Observable que emite los datos solicitados cuando se recibe la respuesta. El tipo de retorno varía según los valores de observe y responseType que pasa a la llamada.

El método get () toma dos argumentos; la URL del punto final desde la que buscar y un objeto de opciones que puede usar para configurar la solicitud.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Las opciones importantes incluyen las propiedades observe y responseType.

* La opción de observación especifica la cantidad de respuesta que se devolverá.
* La opción responseType especifica el formato en el que se devuelven los datos.

Para obtener este tipo de datos, la llamada get () necesita las siguientes opciones: {observe: 'body', responseType: 'json'}. Estos son los valores predeterminados para esas opciones, por lo que los siguientes ejemplos no pasan el objeto de opciones. Las secciones posteriores muestran algunas de las posibilidades de opciones adicionales.

El ejemplo se ajusta a las mejores prácticas para crear soluciones escalables mediante la definición de un servicio inyectable reutilizable para realizar la funcionalidad de manejo de datos. Además de obtener datos, el servicio puede posprocesar los datos, agregar manejo de errores y agregar lógica de reintento.

ConfigService obtiene este archivo mediante el método HttpClient.get ().

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

ConfigComponent inyecta ConfigService y llama al método de servicio getConfig.

Debido a que el método de servicio devuelve un Observable de datos de configuración, el componente se suscribe al valor de retorno del método. La devolución de llamada de suscripción realiza un posprocesamiento mínimo. Copia los campos de datos en el objeto de configuración del componente, que está enlazado a datos en la plantilla del componente para su visualización.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Solicitar una respuesta escrita

Puede estructurar su solicitud HttpClient para declarar el tipo de objeto de respuesta, para que el consumo de la salida sea más fácil y obvio. La especificación del tipo de respuesta actúa como una aserción de tipo en tiempo de compilación.

Para especificar el tipo de objeto de respuesta, primero defina una interfaz con las propiedades requeridas. Utilice una interfaz en lugar de una clase, porque la respuesta es un objeto simple que no se puede convertir automáticamente en una instancia de una clase.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

A continuación, especifique esa interfaz como el parámetro de tipo de la llamada HttpClient.get () en el servicio.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Subtema 11.3:**

Configurar HTTP URL parameters.

**Ejemplos:**

Use la clase HttpParams con la opción de solicitud params para agregar cadenas de consulta de URL en su HttpRequest.

En el siguiente ejemplo, el método searchHeroes () busca héroes cuyos nombres contienen el término de búsqueda.

Comience importando la clase HttpParams.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Si hay un término de búsqueda, el código construye un objeto de opciones con un parámetro de búsqueda codificado en URL HTML. Si el término es "gato", por ejemplo, la URL de solicitud GET sería api / heroes? Name = cat.

El objeto HttpParams es inmutable. Si necesita actualizar las opciones, guarde el valor devuelto del método .set ().

También puede crear parámetros HTTP directamente desde una cadena de consulta utilizando la variable fromString:



**Subtema 11.4:**

Envío de datos a un servidor (POST, PUT, DELETE).

**Ejemplos:**

Además de obtener datos de un servidor, HttpClient admite otros métodos HTTP como PUT, POST y DELETE, que puede utilizar para modificar los datos remotos.

La aplicación de muestra para esta guía incluye una versión abreviada del ejemplo "Tour of Heroes" que busca héroes y permite a los usuarios agregarlos, eliminarlos y actualizarlos. Las siguientes secciones muestran ejemplos de los métodos de actualización de datos del HeroesService del ejemplo.

Hacer una solicitud POST

Las aplicaciones a menudo envían datos a un servidor con una solicitud POST al enviar un formulario. En el siguiente ejemplo, HeroesService realiza una solicitud HTTP POST al agregar un héroe a la base de datos.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

El método HttpClient.post () es similar a get () en que tiene un parámetro de tipo, que puede usar para especificar que espera que el servidor devuelva datos de un tipo determinado. El método toma una URL de recurso y dos parámetros adicionales:

* body: los datos para POST en el cuerpo de la solicitud.
* options: un objeto que contiene opciones de método que, en este caso, especifican los encabezados necesarios.

El ejemplo detecta errores como se describe arriba.

HeroesComponent inicia la operación POST real suscribiéndose al Observable devuelto por este método de servicio.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Cuando el servidor responde con éxito con el héroe recién agregado, el componente agrega ese héroe a la lista de héroes que se muestra.

Hacer una solicitud DELETE

Esta aplicación elimina un héroe con el método HttpClient.delete pasando la identificación del héroe en la URL de la solicitud.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

HeroesComponent inicia la operación DELETE real suscribiéndose al Observable devuelto por este método de servicio.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

El componente no espera un resultado de la operación de eliminación, por lo que se suscribe sin una devolución de llamada. Aunque no esté utilizando el resultado, aún debe suscribirse. Llamar al método subscribe () ejecuta el observable, que es lo que inicia la solicitud DELETE.

Hacer una solicitud PUT

Una aplicación puede enviar solicitudes PUT mediante el servicio de cliente HTTP. El siguiente ejemplo de HeroesService, como el ejemplo POST, reemplaza un recurso con datos actualizados.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En cuanto a cualquiera de los métodos HTTP que devuelven un observable, la persona que llama, HeroesComponent. update () debe suscribe() al observable devuelto por HttpClient.put () para iniciar la solicitud.

**Actividad:**

a) CUESTIONARIO TÉCNICO

* ¿Qué es HttpClientModule?
* Cree un nuevo proyecto en Angular consuma una API Rest utilizando el método HTTP GET.
* Consuma una API Rest utilizando el método HTTP POST.
* Consuma una API Rest utilizando el método HTTP PUT.
* Consuma una API Rest utilizando el método HTTP DELETE.

b) CONCLUSIONES DE LA EXPERIENCIA

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