En este ejemplo vamos a estudiar cómo hacer peticiones AJAX en Angular, para que PHP nos devuelva desde el servidor una cadena JSON .

## Creamos nuestra aplicación

-El primer paso es, usando el CLI, crear nuestra nueva aplicación . Ejecutamos:

`ng new ajax --no-standalone` … , todo por defecto …

Cambiamos al directorio creado …

## Creamos un nuevo componente y un nuevo servicio.

El componente va a contener un botón que llamará a un método propio que a su vez usará (a través de la inyección de dependencia) el servicio que realizará la petición AJAX. Finalmente devolverá el dato en un contenedor DIV del componente.

Creamos el componente ...

```
ng g component botonajax
```

Creamos el servicio …

```
ng g service serviajax
```

No ha sido necesario haber creado un nuevo módulo ya que el ejemplo es muy sencillo, todos los componentes están agregados al módulo principal. Decir que el servicio no es necesario agregarlo al módulo porque es inyectable a nivel raíz '**root**' (global), solo lo importará el componente que lo necesite.

## Configuramos las importaciones necesarias…

Comenzamos con **app.module.ts** , necesitamos importar en él, el siguiente proveedor del servicio y añadirlo a **providers[ ]**:

```
import { provideHttpClient } from '@angular/common/http';
```

(para que el servicio pueda hacer uso del elemento **HttpClient** necesario para la petición AJAX.)

En la plantilla HTML del componente principal **app.component.html** vamos a reemplazar el contenido por nuestro componente **botonajax**:

```
<app-botonajax></app-botonajax>
```

En el servicio **serviajax.service.ts** añadimos :

```
import {HttpClient} from '@angular/common/http';
import { Observable } from 'rxjs';
```

Inyectamos **HttpClient** :

```
constructor(private http: HttpClient) { }
```

Y creamos un método llamado **llamaAjax()** con el siguiente contenido, que se va a encargar de solicitar la petición al archivo PHP devolviendo un observable:

```
llamaAjax(url :string):Observable<any>{
    return this.http.get(url)
}
```

El archivo PHP ya lo conocemos :

```
<?php
// jsonGET.php
// Para solicitudes de otros dominios.
header("access-control-allow-origin: *");
//.....................................
```

```php
$nombre = $_GET['nombre'];

$ciudad = $_GET['ciudad'];

// Devuelve JSON

echo '{"nombre":"'.$nombre.'","ciudad":"'.$ciudad.'"}';

?>
```

Pasamos a configurar nuestro componente **botonajax** , **botonajax.component.ts** , importamos el servicio :

```
import { ServiajaxService } from '../serviajax.service';
```

Inyectamos :

```
constructor(private serviAjax: ServiajaxService ) { }
```

Y creamos el siguiente método, que se subscribe al observable. Habría que adaptar la **url**, puede ser una ruta relativa pero solo funcionaría al desplegar después del compilado (debe estar funcionando el **xamp** para interpretar el PHP) :

```
llamaAjaxC():void{

   this.serviAjax.llamaAjax('http://localhost/tema14/ajax/public/jsonGET.php?nombre=Juan&ciudad=Ubrique')

     .subscribe(data=> {

    var i=document.getElementById('datos') as HTMLDivElement;

     if(i!=null){

         i.innerHTML=`Desde servidor ${data['nombre']} de ${data['ciudad']}`

     };

   });

 }
```

La plantilla de este componente tendrá el siguiente contenido :

```
<button (click)='this.llamaAjaxC()'>Solicitud AJAX</button>
<div id='datos'></div>
```

Probamos: `ng serve -o` , y luego compilamos con **build**, y desplegamos.
***Resumiendo, el botón envía por GET los parámetros en la URL , los recoge el archivo PHP y los devuelve al contenedor del cliente.***

Continuando …, hemos comprobado que aunque la petición ajax funciona en el servicio llamando al método **llamaAjax()**, observamos que estamos accediendo al elemento DIV mediante **document.getElementById('datos')**, esto es válido cuando queremos acceder a elementos en JS , pero Angular tiene sus reglas y acceder directamente al DOM **no es buena práctica**, por tanto debemos usar los mecanismos que nos brinda Angular para no provocar conflictos al manipular el DOM y que nuestro código sea compatible para otros entornos que no sea solo el navegador de escritorio (móviles, web workers, etc..) .
Para acceder al DOM, Angular usa (entre otros) las siguientes clases, servicios y decoradores :

- Clase **Renderer2** que contiene propiedades y métodos para crear, modificar elementos, atributos, clases etc…
  Más info: https://angular.io/api/core/Renderer2#renderer2
- EL decorador **@ViewChild** , para apuntar al identificador del elemento de la plantilla ( ID ) Más info: https://angular.io/api/core/ViewChild
- Los **Observables (librería RxJS)** , que mejoran el rendimiento cuando se tiene que actualizar información recibida desde una operación asíncrona o en el manejo de eventos. Incluso con mejor rendimiento que el enlace de datos bidireccional. Más info: https://angular.io/guide/observables
- **ElementRef** , tipo para declarar elementos del DOM.

## Modificamos nuestro código e importamos los elementos necesarios para realizar una mejora en la petición AJAX.

La clase del servicio **serviajax.service.ts** la dejamos como está …

Pasamos al componente **botonajax.component.ts**

- Importamos los elementos necesarios :

```
import {OnInit ,Renderer2 ,ViewChild, ElementRef,} from
'@angular/core';
```

- Añadimos la declaración del decorador en la clase para que apunte al elemento DIV , por ejemplo por encima del constructor:

```
@ViewChild("datos") idDiv: ElementRef|any;
```

- Inyectamos la clase **Renderer2** :

```
constructor(private serviAjax: ServiajaxService, private rd:
Renderer2) { }
```

- Sustituimos el contenido del método **llamaAjaxC( )** por este nuevo :

```
llamaAjaxC():void{

this.serviAjax.llamaAjax('http://localhost/tema15Angular/ajax/src/app/jsonGET.php?nombre=Juan&ciudad=Ubrique')
      .subscribe(data=>{
        this.rd.setProperty(this.idDiv.nativeElement, 'innerHTML',`Desde servidor ${data['nombre']} de
${data['ciudad']}`)

        }) ;
 }
```

Como dijimos anteriormente, el componente se encarga de renderizar los datos obtenidos del servicio. Usa el método **subscribe()** del observable devuelto por éste último.

Finalmente modificamos la plantilla del componente **botonajax.component.html** añadiendo el selector de elementos dentro del

componente , el cual es usado por el decorador **@ViewChild** , es como el identificador ID (que podríamos omitir).

```
<div #datos id='datos'></div>
```

Reiniciamos servidor y comprobamos …, luego compilamos.