CONOCIMIENTOS VARIOS DE APLICACIONES WEB

DIGITALGEKO

CONOCIMIENTOS VARIOS DE APLICACIONES WEB

Archivos Json

Generaciòn de Tokens

Variables de Sesion

ARCHIVOS TIPO JSON

- ***Un archivo JSON (JavaScript Object Notation) es un formato de texto ligero para almacenar y transmitir datos estructurados, utilizando una sintaxis de clave-valor, legible tanto para humanos como para máquinas.***

JSON es un formato de texto que se utiliza para representar datos estructurados basados en la sintaxis de objetos de JavaScript. Es un formato estándar utilizado para el intercambio de datos entre aplicaciones, especialmente entre servidores y navegadores web.

El formato JSON es fácil de entender y trabajar, y es compatible con la mayoría de los lenguajes de programación, lo que lo convierte en una excelente opción para el intercambio de datos.

SINTAXIS BÀSICA DE JSON

```
{
  "nombre": "Juan",
  "edad": 30,
  "ciudad": "Madrid",
  "hobbies": ["leer", "fútbol", "cine"]
}
```

En este ejemplo:

- Los **nombres de las claves** son cadenas de texto.
- Los **valores** pueden ser cadenas, números, booleanos, objetos, arrays, o null.

MEJORES PRACTICAS

Es importante seguir algunas mejores prácticas para garantizar que el intercambio de datos sea eficiente, seguro y fácil de mantener:

- **Usar la notación correcta**: Asegúrate de que todos los objetos estén bien formados. Cada clave debe estar entre comillas dobles y los valores deben ser del tipo adecuado (cadena, número, array, objeto, booleano o null).

- **Evitar datos innecesarios**: No envíes más datos de los necesarios. Cada dato adicional aumenta el tamaño del archivo, lo que puede afectar el rendimiento.

- **Usar la codificación UTF-8**: Esto garantiza que los caracteres especiales sean tratados correctamente, especialmente si estás trabajando con diferentes idiomas.

- **Estandarizar el formato de fecha**: Si incluyes fechas, utiliza un formato estándar.

- **Validación de JSON**: Siempre valida tu JSON antes de procesarlo para evitar errores. Herramientas como **jsonlint.com** permiten validar la estructura de un archivo JSON.

- **Minimizar el tamaño**: Si el archivo JSON es muy grande, considera comprimirlo o usar una versión minimizada sin espacios y saltos de línea innecesarios.

USOS COMUNES DE LOS ARCHIVOS JSON

- **Intercambio de datos entre cliente y servidor**
- **APIs RESTful**
- **Almacenamiento local (LocalStorage)**
- **Configuración y archivos de preferencia**
- **Bases de datos NoSQL (MongoDB)**
- **Serialización de objetos en aplicaciones**

VENTAJAS Y DESVENTAJAS

Ventajas

- **Ligereza**: JSON es más liviano que otros formatos como XML, lo que reduce la cantidad de datos enviados.
- **Facilidad de uso**: Se puede leer fácilmente tanto por máquinas como por humanos.
- **Compatibilidad**: Es ampliamente compatible con la mayoría de los lenguajes de programación y plataformas.
- **Rendimiento**: Es más rápido de procesar que otros formatos de datos, como XML.

Desventajas

- **Falta de soporte para comentarios**: JSON no permite comentarios, lo que puede ser un inconveniente al tratar de agregar explicaciones a los datos.
- **Menos robusto que XML**: Aunque JSON es más simple, no tiene algunas características avanzadas que ofrece XML, como la validación de esquema o la posibilidad de usar atributos.

TOKENS

- Un **token** es una cadena de caracteres que actúa como un identificador o clave de acceso en el contexto de la autenticación, autorización o transmisión de información entre sistemas. Los tokens son comúnmente utilizados en aplicaciones web, servicios API y sistemas de autenticación para asegurar que solo los usuarios o sistemas autorizados puedan acceder a ciertos recursos.

TIPOS DE TOKENS

- **Token de autenticación**: Usado para verificar la identidad del usuario. Este tipo de token se utiliza generalmente después de que un usuario se haya autenticado con éxito

- **Token de autorización**: Después de la autenticación, un token también puede usarse para verificar los permisos del usuario, como si tiene acceso o si está autorizado para realizar una acción.

- **Token de acceso**: Es un token utilizado para autenticar solicitudes a un servidor. En sistemas de autenticación basados en tokens como OAuth2, el **token de acceso** es el que permite a los usuarios acceder a los servicios protegidos.

- **Token JWT (JSON Web Token)**: Un tipo de token basado en JSON que es comúnmente utilizado en aplicaciones web modernas para la autenticación y autorización. Este token contiene información (como la identidad del usuario y sus permisos) y se firma de forma segura.

MEJORES PRÀCTICAS

- **Usar HTTPS para la transmisión de tokens**
- **No almacenar tokens en lugares inseguros**
- **Usar tokens con tiempo de expiración (expiración corta)**
- **Implementar refresh tokens**
- **Firmar los tokens adecuadamente**
- **Evitar exponer datos sensibles en el payload del token**
- Revocar tokens cuando sea necesario
- **Limitar el acceso a la API y usar scopes**

VARIABLES DE SESIÒN

- Es un **tipo de almacenamiento** que se utiliza para mantener la **información del usuario** durante su interacción con una aplicación web o un sitio. Estas variables se **almacenan en el servidor** y se **asocian a una sesión específica del usuario,** lo que permite que los datos se mantengan a lo largo de múltiples solicitudes del mismo usuario.

CARACTERÌSTICAS

- **Almacenamiento en el servidor**: Las variables de sesión se almacenan en el servidor, no en el navegador del cliente (como ocurre con las cookies).
- **Persistencia temporal**: La información contenida en una variable de sesión persiste durante el tiempo que dure la sesión del usuario. Cuando la sesión expira (por ejemplo, por inactividad) o el usuario cierra el navegador, las variables de sesión se destruyen.
- **ID de sesión**: Cada usuario tiene un identificador único de sesión (generalmente en forma de cookie), que el servidor utiliza para rastrear la sesión del usuario y acceder a las variables asociadas a esa sesión.

COMO FUNCIONAN

1. El servidor crea una sesión única para cada usuario cuando éste visita la aplicación.
2. El servidor asigna un identificador único (ID de sesión) a esa sesión y lo envía al navegador del usuario como una cookie.
3. El servidor almacena las variables de sesión asociadas al ID de sesión. Estas variables pueden contener cualquier tipo de información relevante (nombre de usuario, preferencias, autenticación, etc.).
4. En cada solicitud posterior del usuario, el navegador envía el ID de sesión, permitiendo al servidor recuperar la información almacenada en las variables de sesión y seguir interactuando con el usuario sin tener que pedirle que se autentique nuevamente.

MEJORES PRÀCTICAS

- **Usar session_start() en cada página que utilice sesiones**
- Proteger el ID de sesión
- Limitar la duración de la session
- Almacenar solo información necesaria en la session
- **Cerrar la sesión de manera adecuada**
- Utilizar sesiones en combinación con tokens de autenticación
- Validar la información de la sesión regularmente