DESARROLLO WEB EN ENTORNO CLIENTE

CAPÍTULO 8:

Almacenamiento de datos en el lado cliente

Juan Manuel Vara Mesa
Marcos López Sanz
David Granada
Emanuel Irrazábal
Jesús Javier Jiménez Hernández
Jenifer Verde Marín

Ra-Ma®

Objetivos

- Mecanismos de almacenamiento Web del lado del cliente.
- La especificación Web Storage de la W3C.
- Los objetos de almacenamiento Web de HTML 5 e IndexedDB.
- Los conceptos genéricos de las bases de datos del lado del cliente.

Objetivos

- Uno de los pilares de la personalización se encuentra en el concepto de sesión y en la habilidad de almacenar datos del usuario que utiliza el sitio Web en el mismo navegador cliente con el fin de mejorar la experiencia del usuario.

Objetivos

- Si la información pertenece al usuario debe estar en la localización del usuario (es decir en el navegador cliente). A continuación veremos las diferentes opciones de almacenamiento de datos en los navegadores clientes.

Las Cookies

- Una cookie es sólo información en texto plano administrable por el mismo usuario y en ningún caso es código fuente interpretable. Parte de la sencillez de las cookies genera inconvenientes:
  - Cada navegador tendrá sus propias cookies.
  - Las cookies no diferencian entre usuarios que utilicen el mismo navegador en una misma sesión del sistema operativo. Muchas veces queda almacenada la información de nuestra tarjeta de crédito al realizar una transferencia bancaria.
  - Son vulnerables a los “sniffer” (programas que pueden leer el contenido de peticiones y respuestas HTTP) debido a que estas se realizan en texto plano.
  - Las cookies pueden ser modificadas en el cliente, lo cual podría aprovechar vulnerabilidades del servidor.

Navegadores que implementan Web Storage

- HTML 5 incluye dos nuevos objetos para el almacenamiento de datos en el cliente: los “sessionStorage” y los “localStorage”.

Característica	Internet Explorer	Firefox	Chrome	Opera	Safari
localStorage	8	3.5	4	10.50	4
sessionStorage	8	2	5	10.50	4

Web Storage

- Esta especificación introduce dos mecanismos relacionados para obtener la persistencia de datos de manera estructurada del lado del cliente, similares al mecanismo de las cookies.

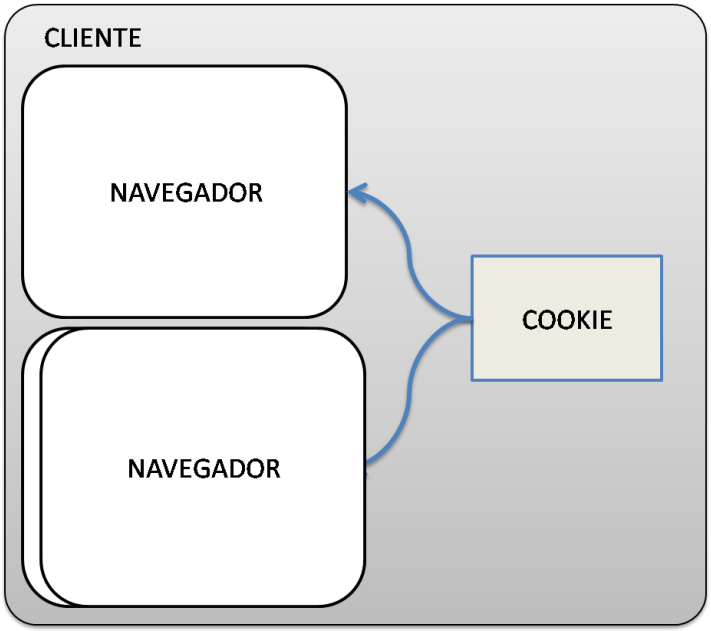
  - El contenido de las cookies es enviada al servidor en cada petición.
  - En HTML 5 la información solo podrá ser accedida desde el lado del cliente.
  - Es posible almacenar gran cantidad de información sin afectar el rendimiento de la aplicación Web.

Web Storage: escenarios de uso

- Almacenamiento de datos.
- Utilización fuera de línea.
- Mejora de rendimiento.
- En el caso del “sessionStorage” no existe relación entre lo almacenado en diferentes pestañas o ventanas de un mismo navegador.
- Con el objeto “sessionStorage” existe la seguridad de que los datos serán borrados una vez termine la sesión de la ventana que lo ha utilizado.

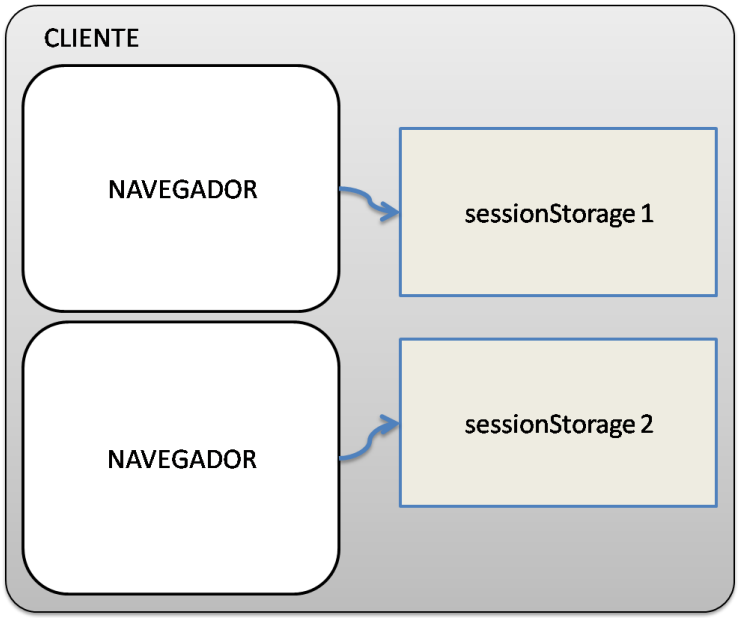
SessionStorage

- Problema: múltiples transacciones en diferentes ventanas o pestañas de un navegador.
- En caso de que la aplicación utilice cookies para mantener el estado.
- Todas las ventanas están asociadas a una misma sesión.



SessionStorage

- La especificación actual de Web Storage incluye un nuevo objeto llamado “sessionStorage”.
- El objeto “sessionStorage” se instancia por sesión y ventana, por lo que dos pestañas del navegador abiertas al mismo tiempo y para un mismo sitio Web pueden tener información distinta.
- Al cerrar la sesión se pierde la información.



Métodos del objeto SessionStorage

- Agregar un nuevo par clave/valor:

```
sessionStorage.setItem("maleta","1");
```

- Obtener el valor en base a la clave:

```
var item = sessionStorage.getItem("maleta");
```

- Remover el par clave/valor:

```
var item = sessionStorage.removeItem("maleta");
var item = sessionStorage.removeItem(1);
```

- El método "clear()" borra todos los elementos de la lista:

```
sessionStorage.clear();
```

LocalStorage

- Este objeto se extiende a lo largo de múltiples ventanas y múltiples sesiones.
- Puede ser accedido cada vez que se visita el dominio (los subdominios no son válidos) y todas las sesiones abiertas sobre la misma Web ven la misma información.

- Cualquier tipo de cambio en el almacén debe disparar un evento de tipo "StorageEvent", de forma que cualquier ventana con acceso al almacén pueda responder al mismo.

Propiedades del evento “storage”

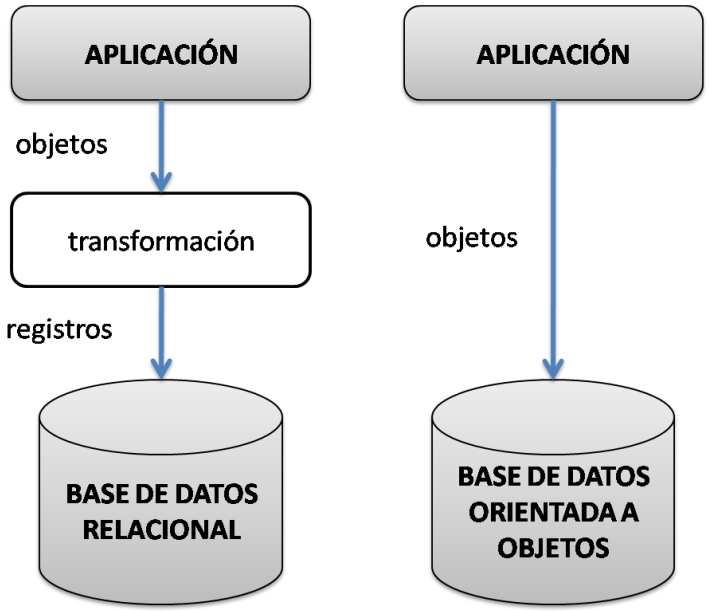
Propiedades	Internet Explorer
`url`	El dominio asociado con el objeto que ha cambiado.
`storageArea`	Representa el objeto localStorage o sessionStorage afectado.
`key`	La clave del par clave/valor que ha sido agregado, modificado o borrado.
`newValue`	El nuevo valor asociado con la clave. Será null si trata de una eliminación.
`oldValue`	El antiguo valor.

Persistencia de los datos

- Dependiendo el navegador, existirán diferentes estrategias de implantación para persistir los datos.
- Firefox realiza el almacenamiento de manera síncrona, persiste inmediatamente los datos que se almacenan mediante código.
- En Internet Explorer los datos se escriben de manera asíncrona.
- En Internet Explorer 8 se puede forzar la persistencia mediante los métodos “begin” y “commit”. Las asignaciones de datos que se encuentren encerrados entre la llamada de ambos métodos se realizarán de manera síncrona.

Bases de Datos Orientadas a Objeto

- Las aplicaciones se encuentran mayormente desarrolladas a partir de la orientación a objetos.
- Las bases de datos relacionales no están preparadas para almacenar objetos sino para almacenar registros.
- En las bases de dato orientadas a objeto se realiza la persistencia de los objetos utilizados en la programación de manera transparente, sin necesidad de transformar los datos.



Base de datos en el cliente

- En HTML 5 se especifican las características de un sistema de base de datos de objeto en el lado del cliente.
- Base de datos: un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.
- La base de datos tiene una estructura, lo que permite mantener una mayor eficiencia en los datos y la aplicación de búsquedas directas.
- Las bases de datos locales permiten almacenar datos en el cliente teniendo acceso a esta información sin necesidad de estar en línea.

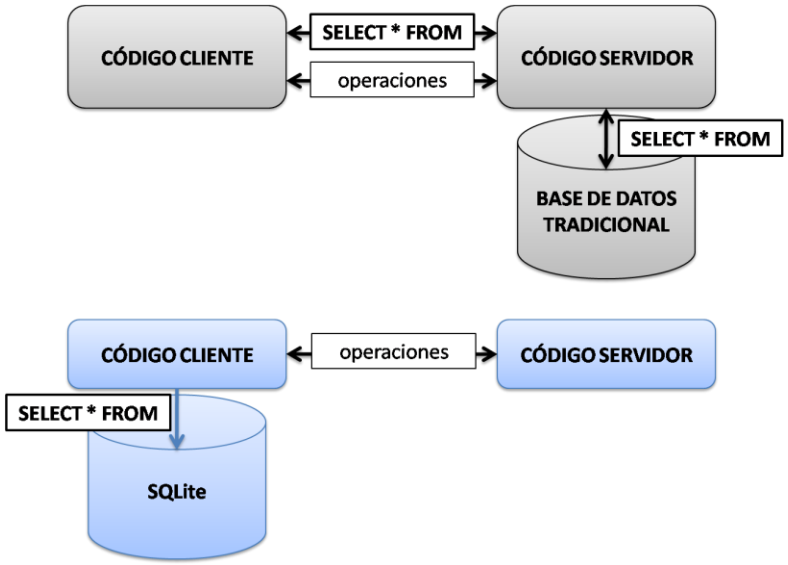
Base de datos en el cliente

- En las aplicaciones Web existen principalmente 2 técnicas de bases de datos locales:
  - WebSQL (basado en SQLite): que actualmente no tiene más soporte pero que ha sido y es utilizado por los navegadores.
  - Indexed Database API: impulsado por la W3C y Oracle es la propuesta de estándar actualmente vigente. Se encuentra parcialmente implementado en las últimas versiones de los navegadores.

WebSQL

- Está basada en SQLite, un gestor de base de datos relacionales construida en el lenguaje C.

- Se enlaza directamente con la lógica de negocio de la aplicación (en este caso, por ejemplo con el código del lado del cliente).



Base de datos en el cliente

- IndexedDB es una interfaz de aplicación de programas (API) para el almacenamiento de gran cantidad de datos de manera estructurada utilizado en HTML 5:
  - IndexedDB almacena directamente objetos, por lo tanto, es posible persistir objetos JavaScript.
  - Para ello hace uso de un mecanismo denominado “Object Store” para lograr la persistencia.

Comparación BBDD relacional vs IndexedDb

Característica	Base de datos relacional	Indexed Database API
Tabla.	La tabla contiene filas y columnas.	Los "Object Store" o Contenedores que contienen objetos Javascripts y claves.
Consultas.	SQL.	Cursores y rangos de clave. Son necesarios los índices para realizar las búsquedas.
Transacciones.	De lectura/escritura y a nivel de tablas o registros.	Read_Version_Change, Read_Write, Read_Only.
Estructura de los datos.	Cumple con las reglas de forma normal.	Estructura no normalizada.

Contenedor en IndexedDb

- En IndexedDB no se trabaja con el concepto de consultas SQL, en cambio se trabaja con índices y cursores que recorren los objetos.
- El “Object Store” o “Contenedor” almacena pares clave/valor.
- Los valores por lo general serán objetos con una estructura compleja.
- La clave podrá ser una propiedad de este objeto.
- La búsqueda se puede realizar a partir de propiedades del mismo objeto.

Claves en IndexedDb

- Clave: un valor a partir del cual los objetos almacenados son organizados.

- En cada Contenedor no se puede repetir la clave.

- Características:
  - Un generador de claves: con el cual se producen nuevas claves de forma artificial y ordenada.
  - In-line Key: la clave es almacenada como parte del valor.
  - out-of-line key: la clave es almacenada fuera del valor almacenado.
  - key path: define el lugar desde donde el navegador extrae la clave en el Contenedor.

Crear la Base de Datos

- Para la gestión de la base de datos es indispensable el método “open”.
- Este método funciona de dos maneras: crea la base de datos si esta no existe o la abre en caso de que exista.
- Si todo funciona correctamente el evento “success” es disparado y la función indicada en la propiedad “onsuccess” es aplicada.
- En caso existir un error se dispara la función relacionada con la propiedad “onerror” de la petición con el mismo evento de error como argumento.

```
var peticion = window.indexedDB.open(“nombre”);
peticion.onsuccess = respuesta_exito;
peticion.onerror  = respuesta_error;
```

Eliminar la Base de Datos

- Uno de los atributos más importantes de la base de datos es la versión, que sirve como identificador único. La aplicación Web podrá acceder en todo momento a una única versión de la base de datos. Este atributo es una cadena de caracteres inicializada al valor vacío una vez creada la base de datos.

```
var peticion = bbdd.setVersion("VERSION_1");
```

- La eliminación de una base de datos es similar a la creación.

```
var peticion = window.indexedDB.deleteDatabase("nombre");
```

Creación de Contenedores e Índices

- El contenedor creado en el ejemplo es “contactos” y su “keyPath” será la variable “tel”. Es decir que los objetos incluidos en el Contenedor tendrán un valor de nombre “tel” que también será la clave.
- Se ha creado un índice mediante la función “createIndex”. Se incluye el nombre del índice y el valor asociado al índice (en este caso será la variable “correo”).
- Por último se realiza la carga de objetos en el Contenedor.

```
contacto = [{tel: “1234”, correo:“juan@mail.com”, movil:“666123”},
            {tel: “2345”, correo:“pedro@mail.com”, movil:“664392”}];

peticion.onsuccess=function(event)
 {
  var contenedor = bbdd.createObjectStore(“contactos”, {keyPath:“tel”});

  contenedor.createIndex("correo",  “correo”, {unique:true});

  for (n in contacto){contenedor.add(contacto[n]);}
 };
```

Transacciones

- Las operaciones con las que se agregarán, modificarán y eliminarán datos debemos realizarlas en una transacción.
- En base de datos se indica como transacción a un conjunto de operaciones que se aplican formando una unidad de tal manera que dentro de la transacción se mantiene la integridad de los datos, haciendo que estas operaciones no puedan finalizar en un estado intermedio.

```
 var transaccion = bbdd.transaction(["contactos"],
IDBTransaction.READ_WRITE);

 transaccion.oncomplete = function(evento){…};
 transaccion.onabort = function(evento){…};
 transaccion.onerror = function(evento){…};
```

Operaciones

- Una vez que tenemos la transacción abierta podemos operar con los Contenedores y objetos. Para agregar un nuevo elemento utilizamos la función “add”. El resultado de agregar un nuevo objeto (“evento.result”) es la clave de dicho objeto agregado.

- No es posible agregar dos objetos con la misma clave mediante la operación “add”. En lugar de ello, para modificar un elemento existente utilizamos la función “put”.

- Para eliminar un elemento utilizamos la función “delete”. En el siguiente ejemplo llamamos al Contenedor directamente a partir de la transacción y borramos el objeto cuya clave es “1234”.

Obtención de datos

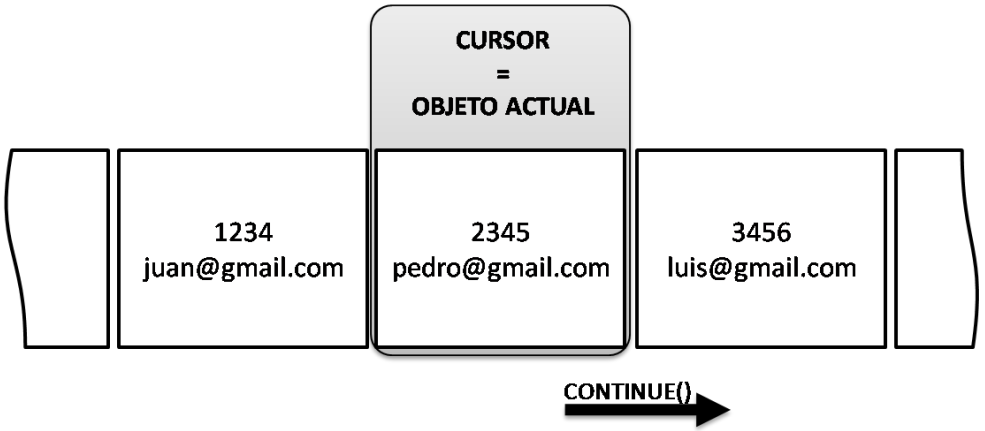
- La API permite obtener tanto objetos simples como conjunto de objetos. Para el caso de los objetos simples se utiliza el método “get” con la clave del objeto a ser recuperado.

- En el caso de que busquemos obtener más de un objeto es necesario utilizar un cursor para recorrerlos, limitando o no el rango de objetos.

El Cursor

- El cursor es el resultado del evento en caso de éxito (es decir, se encuentra en la propiedad “result”.

- En este caso el cursor no tiene ningún parámetro adicional por lo que recorrerá los contactos hasta finalizar. En la variable cursor se encuentra el objeto actual obtenido, de tal manera que con el método “continue” se carga la variable con el siguiente objeto.



Aplicaciones en CACHÉ

- La caché de las aplicaciones Web es el proceso de almacenar datos generados dinámicamente para que puedan ser utilizados nuevamente sin necesidad de realizar peticiones al servidor.

- Es utilizado principalmente para mejorar el rendimiento de las aplicaciones Web y disminuir el tiempo de carga.

Ventajas y Desventajas

- Dos ventajas principales:
  - El rendimiento en las aplicaciones Web es un aspecto fundamental. En ese sentido cualquier mejora provoca un cambio significativo en la experiencia del usuario.
  - Es una tecnología ampliamente extendida y por lo tanto los navegadores Web implementan componentes de caché utilizables por las aplicaciones.
- Dos desventajas principales:
  - En algunos escenarios (por ejemplo en aplicaciones ya construidas) puede ser complejo agregar aspectos de caché.
  - El comportamiento de la aplicación puede verse afectado, especialmente si se realiza una mala programación.

Caché de Aplicación en HTML 5

- El objetivo de estas nuevas características, junto con la API de IndexedDB es la posibilidad de obtener aplicaciones Web y sitios Web que funcionen correctamente cuando no tienen la conexión con el servidor Web (modo fuera de línea).

- El primer paso para realizar la caché de aplicación es la elaboración de un fichero llamado MANIFEST (o manifiesto en español). En este fichero listaremos el conjunto de recursos con los que se trabajará de modo fuera de línea.

Caché de Aplicación en HTML 5

- Comenzará con la cadena: “CACHE MANIFEST” y contendrá por lo menos tres partes:

  - **CACHE**: las URLs que se encuentran en esta sección serán mantenidas en la caché de la aplicación por el navegador para ser vistas en modo fuera de línea.

  - **NETWORK**: las URLs de esta sección no serán cargadas a la caché de aplicación. Es común utilizar el carácter comodín asterisco “*”.

  - **FALLBACK**: indica el contenido que será mostrado en caso de que un recurso no sea encontrado, por ejemplo cuando la aplicación se encuentra fuera de línea y es necesario cargar un recurso que se encuentra en la sección NETWORK.

Caché de Aplicación en HTML 5

```
CACHE MANIFEST

CACHE:
index.html
pagina.css
pagina.js

NETWORK:
pago.html
ultimasnoticias.html

FALLBACK:
offline.html
```

Referenciar el Manifiesto

- Una vez creado el manifiesto debemos referenciarlo dentro de la etiqueta HTML de las páginas HTML que estarán en la caché de aplicación.

```
<html manifest="principal.appcache">
  …
</html>
```

Mantener la Integridad

- Al visitar el sitio Web por primera vez se descargan los recursos que aparecen en el manifiesto.

- **Problema**: al pasar del modo fuera de línea al modo en línea se ha actualizado en el servidor algún recurso que se encuentre en la caché de aplicación.

Se sigue mostrando la versión antigua contenida en la caché de aplicación.

- La **solución** es actualizar el manifiesto. Y para no modificar el contenido del fichero recomendamos incluir un comentario con la versión o fecha de última actualización del manifiesto.