PAU CASESNOVES
Centre Integrat de Formació Professional

Desenvolupament d'aplicacions web

Desenvolupament web a l'entorn client

UT 5.3: Guardar informació al client

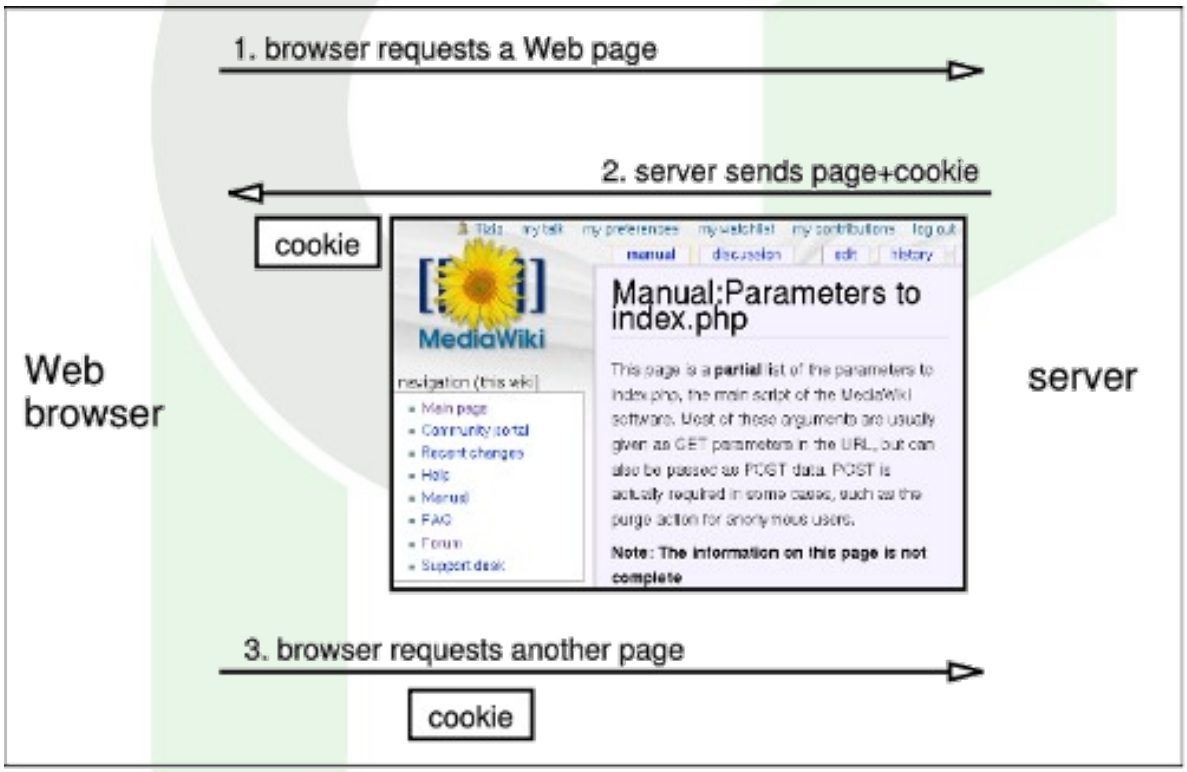
Joan Pons Tugores

Índex de continguts

Cookies

Una cookie és una variable que es guarda a l'ordinador del client, al seu navegador. Cada vegada que aquest navegador demani una pàgina del mateix servidor, també enviarà la cookie. No s'haurien d'utilitzar per guardar informació que només necessitam al client.

Inicialment les cookies només es podien crear des del servidor, però actualment es poden crear d'altres maneres, per exemple amb javascript.



Els usos més freqüents són:

- Guardar el nom d'usuari i contrasenya per evitar tornar-ho a introduir.
- Mantenir un seguiment de les compres en una botiga virtual.
- Utilitzar opcions de continguts o disseny escollides anteriorment.

- Obtenir informació sobre els hàbits de navegació de l'usuari.

Les galetes han estat sovint un tema recurrent en la privacitat a internet. Degut a la informació que guarden, és relativament fàcil fer un seguiment de les operacions de l'usuari repassant les galetes creades, podent-se crear un perfil d'usuari que permeti reconèixer les seves preferències. Aquests perfils, malgrat ser anònims (llevat que l'usuari introdueixi voluntàriament informació personal), son considerats per alguns com una vulneració de la privacitat.

Malgrat això cal destacar que les galetes són únicament fragments d'informació i que per tant no poden esborrar dades, obrir finestres emergents o generar correu brossa, com a vegades s'ha dit.

Descripció d'una cookie

Ja hem dit que una cookie és una peça d'informació que el servidor guarda al nostre client, normalment amb la finalitat de millorar la navegació per el seu lloc i que s'envien al servidor en cada petició.

Des de la pròpia pàgina HTML podem accedir a les cookies a través de JavaScript. Ho farem a través de l'objecte *document.cookie*.

Per guardar una *cookie* montam una cadena i l'assignam a *document.cookie*.

```
document.cookie=cookie_name+"="+cookie_value
```

Si llegim aquest objecte obtenim una cadena de text amb totes les cookies accessibles amb el format nom=valor; nom=valor;

```
let cadena=document.cookie; //Recuperam les cookies
let galetes=cadena.split(";"); //Obtenim un array nom=valor
for (let i = 0; i < galletes.length; i++) {
    let parts = galletes[i].split("="); //Separam el nom dels valors.
    nom = parts[0];
    valor = parts[1];
```

A cada cookie li podem definir els següents atributs:

```
document.cookie="nom="+valor+";expires=-1;path=/calaixet"
```

- **domain**: indica el domini per el qual s'utilitzarà la cookie. La pàgina ha de pertànyer a aquest domini. Si no posam aquest atribut el valor per defecte és el domini de la pàgina. No podem posar explícitament com a domini localhost.
    - aulavirtual.paucasesnovescifp.cat només a aquest domini
    - .paucasesnovescifp.cat a qualsevol subdomini d'aquest domini

- **path**: determina la ruta dins el domini on s'utilitzarà la cookie.
    - /calaixet: la cookie només s'utilitzarà dins la carpeta calaixet del domini.
- **expires**: indica la data de caducitat de la cookie. Passada aquesta data s'eliminarà del client en accedir al seu domini. Si no apareix aquest atribut caduca en acabar la sessió actual.
- **secure**: si la cookie té aquest atribut les comunicacions per a transmetre aquesta cookie han de ser encriptades.
- **httpOnly**: si la cookie té aquest atribut només es pot utilitzar dins el protocol http. No és pot utilitzar, per exemple, des de JavaScript.

Cookies i Javascript

Aquí teniu tres funcions JavaScript per treballar amb cookies.

- setCookie guarda una nova cookie al navegador, o canvia el seu valor i atributs si ja existeix. Els atributs van separats per un punt i coma i un espai exactament.

```
function setCookie(c_name, value, exdays, domain, path) {
    let exdate = new Date();
    //expirarà d'avui  a exdays dies
    exdate.setDate(exdate.getDate() + exdays);
    let c_value = encodeURI(value)
        + ((exdays == null) ? "" : "; expires=" + exdate.toUTCString())
        + ((domain == null) ? "" : "; domain=" + encodeURI(domain))
        + ((path == null) ? "" : "; path=" + encodeURI(path));
    document.cookie = c_name + "=" + c_value;
}
```

- getCookie torna el valor de la cookie que té el nom que rep com a paràmetre. Si no la pot trobar tornarà *undefined*. Només podem recuperar el nom i el valor, els altres atributs no. Només podem recuperar cookies del domini de la pàgina que executa la funció.

```
function getCookie(nomCookie) {
   let galletes = document.cookie.split(";");
   for (let i = 0; i < galletes.length; i++) {
      let parts = galletes[i].split("=");
      let nom = parts[0];
      let valor = parts[1];
      nom = nom.replace(/^\s+|\s+$/g, ""); // trim del nom
```

```
        if (nom === nomCookie) {
            return decodeURI(valor);
        }
    }
}
```

- deleteCookie elimina la cookie que té el nom que rep com a paràmetre. Per fer-ho li estableix la data d'expiració al valor d'ahir. Per tant el navegador l'eliminarà.

```
function deleteCookie(nom){
    setCookie(nom,"",-1);
}
```

Local storage

Una altra forma de guardar dades al client és el *Local storage*. És un altre mecanisme per guardar parells clau – valor al client, de manera que persisteixin entre sessions.

La principal diferència amb les cookies és que les dades guardades al *Local storage* no s'envien al servidor.

La implementació depèn del navegador, però tant *chrome* com *firefox* utilitzen *SQLite* per guardar aquesta informació.

Local storage i Javascript

Des de JavaScript tenim accés al local storage des d'una propietat només de lectura de l'objecte *window*, *window.localStorage*.

Aquesta propietat ens torna un objecte de tipus *Storage*. L'origen de la pàgina ha de ser una combinació protocol – host - port vàlida. Per a cada una d'aquestes combinacions el navegador guardarà un objecte d'aquest tipus diferent.

Per exemple

```
http://www.paucasesnovescifp.cat:8080
https://www.paucasesnovescifp.cat:8080
```

al client tendrem dos objectes *Storage* diferents.

Normalment els navegadors no utilitzen el *local storage* si l'usuari ha especificat que no volia *cookies*.

A diferència de les cookies, les dades guardades al *local storage* no tenen data de caducitat, es guarden fins que s'esborrin amb *removeItem o clear*.

Mètodes de l'objecte *Storage*

- .**setItem**(clau, valor): afegeix el parell clau – valor a l'objecte. Si ja hi havia un parell amb la mateixa clau, substitueix el valor associat a aquella clau.

```
localStorage.setItem('color','blau'); //Crea el parell color – blau
localStorage.setItem('color','vermell'); //Canvia el valor del color a vermell
```

- .**getItem**(clau): torna el valor associat a la clau. Si la clau no existeix torna *null*.

```
const color = localStorage.getItem('color'); //torna vermell
```

- .**removeItem**(clau): Elimina el parell clau – valor de l'*storage*. Si no existia no fa res.

```
localStorage.removeItem('color'); //Elimina color de l'storage
```

- .**clear**(): Elimina tots els parells clau – valor de l'*storage*.

```
localStorage.clear(); //Elimina tot el de l'storage
```

- .**key**(sencer): Torna el nom de la clau que ocupa la posició *sencer*. Si no existeix torna *null*.

```
localStorage.key(0); //Torna color
```

- .**length**: En realitat és una propietat, no un mètode. Torna la quantitat de parelles clau – valor guardats dins del *storage*.

```
const mida = localStorage.length;
```

Session Storage

És el mateix mecanisme que el *local storage* només que està lligat a la sessió. Quan la sessió es tanca l'*storage* que té associat es destrueix.

Session storage i Javascript

Des de JavaScript tenim accés al *session storage* des d'una propietat només de lectura de l'objecte *window*, *window.sessionStorage*.

Aquesta propietat ens torna un objecte de tipus *Storage*. L'origen de la pàgina ha de ser una combinació protocol – host - port vàlida. Per a cada una d'aquestes combinacions el navegador guardarà un objecte d'aquest tipus diferent.

Per a cada pestanya del navegador, encara que pertanyin al mateix domini, es crea un objecte *session storage* diferent.

L'objecte associat a la pestanya sobreviu fins que la pestanya es tanqui, encara que es recarregui la pàgina.

La única diferència a nivell de programació entre el *session storage* i el *local storage* és la propietat utilitzada per accedir-hi. L'objecte que tornen aquestes propietats són del tipus *Storage* tots dos, per tant tenen els mateixos mètodes.

```
sessionStorage.setItem('color','blau'); //Crea el parell color – blau

const color = sessionStorage.getItem('color'); //torna vermell

sessionStorage.removeItem('color'); //Elimina color de l'storage

sessionStorage.clear(); //Elimina tot el de l'storage
```