Cookie e Webstorage

Un **cookie** (letteralmente *biscotto*) è un pacchetto di informazioni che viene salvato sul computer dell'utente. In successive sessioni di collegamento a Internet (normalmente per un periodo di tempo limitato) il browser può accedere ed utilizzare questi dati.
Utilizzare i *cookie* permette a un sito di ricordare alcune informazioni sull'utente: in alcuni casi può essere molto comodo (per esempio si evita che un utente debba inserire ogni volta il nome utente e la password), ma può generare problemi di sicurezza.
Si possono creare *cookie* in vari modi, per esempio utilizzando il linguaggio *Php* oppure il linguaggio *JavaScript*.

L'utilizzo dei *cookie* presenta due limiti fondamentali:

- La dimensione massima di un *cookie* è di soli 4 Kbyte e un dominio solitamente non può utilizzare più di 20 *cookie*. Se fino a qualche anno fa i *cookie* venivano utilizzati solo per pochi dati (nome utente, ecc.), con lo sviluppo delle applicazioni Web questo limite è diventato troppo restrittivo.
- Non si possono impostare *cookie* diversi per sessioni contemporanee sullo stesso browser. In passato i *browser* non facevano uso di schede per la visualizzazione delle pagine, per cui raramente l'utente poteva tenere aperte più pagine Web contemporaneamente. Ora la disponibilità di banda larga e le capacità dei computer permettono la navigazione contemporanea su decine di pagine Web: l'utente può, per esempio, tenere aperte nello stesso momento le caselle di posta elettronica su due *account* differenti.

Nella versione HTML5 la funzionalità di **Webstorage** permette di memorizzare dati in maniera simile ai *cookie* e risolve i problemi appena visti.
Webstorage comprende due oggetti principali che possono sostituire le funzionalità dei *cookie*:

- **localStorage**: può contenere dati che sono comuni a tutto il dominio (esattamente come i *cookie*). Se vengono aperte due finestre su uno stesso sito, i dati memorizzati da una pagina del sito sono disponibili nell'altra pagina.
- **sessionStorage**: i dati sono disponibili solamente alla pagina che li ha creati.

Per esempio, supponiamo di aprire in due finestre il sito di *Gmail*. Effettuiamo l'accesso con l'account di posta elettronica:

- Se i dati sono memorizzati in *localStorage*, ci si trova automaticamente collegati anche nella seconda finestra. Effettuando il logout nella seconda pagina, il collegamento viene chiuso anche nella prima pagina.
- Se i dati sono memorizzati in *sessionStorage*, si può effettuare il login nella seconda finestra con un account differente.

Questo non significa che il secondo metodo sia migliore del primo, perchè in alcuni casi può essere utile condividere le informazioni da una finestra all'altra. Per esempio, in un sito di *e-commerce*, i prodotti scelti e presenti nel carrello devono essere gli stessi in tutte le finestre.

Vediamo ora i metodi dell'oggetto *localStorage*.

Per memorizzare un valore si utilizza il metodo **setItem**:

```
localStorage.setItem(nomeVariabile, valore);
```

Per richiamare un valore si utilizza il metodo **getItem**:

```
localStorage.getItem(nomeVariabile);
```

Per cancellare un valore si utilizza il metodo **removeItem**:

```
localStorage.removeItem(nomeVariabile);
```

Per cancellare tutti i valori si utilizza il metodo **clear**:

```
localStorage.clear();
```

Metodi analoghi sono disponibili per l'oggetto *sessionStorage:*

```
sessionStorage.setItem(nomeVariabile, valore);
sessionStorage.getItem(nomeVariabile);
sessionStorage.removeItem(nomeVariabile);
sessionStorage.clear();
```

È sempre opportuno prevedere il comportamento della pagina Web nel caso in cui il browser utilizzato non sia compatibile con *Webstorage*. Al caricamento della pagina Web, si può eseguire la verifica:

```
if (typeof(localStorage) == 'undefined' ) {
    alert('Attenzione: il tuo browser non supporta Webstorage.');
}
```

La seguente pagina Web informa se il browser in uso è compatibile con *Webstorage*. La procedura di controllo viene avviata al caricamento della pagina (evento **onload**) senza alcun intervento da parte dell'utente.

```
<!doctype html>
<html lang="it">
<head>
<script type="text/javascript">
function controllo(){
   if (typeof(localStorage) == 'undefined' ) {
      alert('Attenzione: il tuo browser non supporta Webstorage.');
   } else {
      alert('Tutto ok! Il tuo browser supporta Webstorage.');
   }
}
</script>
</head>
<body onload="controllo()">
</body>
</html>
```

ESEMPIO

Gestire i dati inseriti dall'utente con un form.

Il form contiene due campi di tipo testo in cui inserire il nome e il cognome e un pulsante per annullare i dati inseriti e svuotare le caselle. L'utente attiva le operazioni con tre pulsanti per: salvare i dati, mostrare i dati ed eliminare i dati (*Salva*, *Mostra*, *Elimina*). Quando la pagina viene caricata, oppure quando l'utente fa clic su uno dei pulsanti, i dati vengono aggiornati e visualizzati. La gestione dei dati viene fatta con l'oggetto **localStorage**.

Le seguenti righe di codice rappresentano il form HTML.

```
<form>
   <p>
   Nome:<input type="text" id="nome">
   Cognome:<input type="text" id="cognome">
   <input type="reset" value="Annulla" />
   </p>
   <p>
   <input type="button" onclick="salvaDati()" value="Salva" />
   <input type="button" onclick="mostraDati()" value="Mostra" />
   <input type="button" onclick="eliminaDati()" value="Elimina" />
   </p>
</form>
```

Le due caselle di testo hanno id *nome* e *cognome*. Il pulsante *Annulla* svuota le caselle di testo. Dei tre pulsanti sotto le caselle, il primo avvia la funzione *salvaDati*, il secondo la funzione *mostraDati* e il terzo *eliminaDati*.
Alla fine del form è inserito un paragrafo vuoto chiamato *datiSalvati* per visualizzare i dati:

```
<p id="datiSalvati"></p>
```

Nel codice JavaScript della funzione *salvaDati,* le prime due righe memorizzano il contenuto delle caselle di testo nelle variabili *nomeDaSalvare* e *cognomeDaSalvare*. Il metodo **setItem** effettua poi la memorizzazione.

```
function salvaDati() {
   var nomeDaSalvare = document.getElementById("nome").value;
   var cognomeDaSalvare = document.getElementById("cognome").value;
   localStorage.setItem("nome", nomeDaSalvare);
   localStorage.setItem("cognome", cognomeDaSalvare);
   mostraDati();
}
```

La funzione *mostraDati* visualizza i dati in un paragrafo sotto il form: le prime due righe recuperano i dati tramite il metodo **getItem**, mentre l'ultima riga visualizza i dati nel paragrafo chiamato *datiSalvati*.

```
function mostraDati() {
   var nome = localStorage.getItem("nome");
   var cognome = localStorage.getItem("cognome");
   document.getElementById("datiSalvati").innerhtml = nome + " " + cognome;
}
```

Infine la funzione *eliminaDati* richiama il metodo **clear** per cancellare tutti i dati.

```
function eliminaDati() {
   localStorage.clear();
   mostraDati();
}
```

Pagina Web (*gestioneDati.html*)

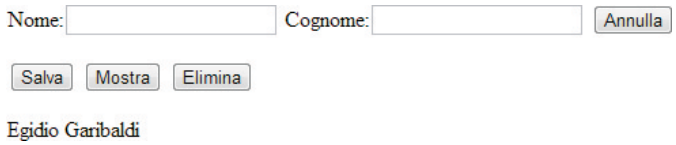
```
<!doctype html>
<html lang="it">
<head>
<script type="text/javascript">
function salvaDati() {
   var nomeDaSalvare = document.getElementById("nome").value;
   var cognomeDaSalvare = document.getElementById("cognome").value;
   localStorage.setItem("nome", nomeDaSalvare);
   localStorage.setItem("cognome", cognomeDaSalvare);
   alert("Dati salvati.");
   mostraDati();
}

function mostraDati() {
   var nome = localStorage.getItem("nome");
   var cognome = localStorage.getItem("cognome");
   document.getElementById("datiSalvati").innerhtml = nome + " " + cognome;
}

function eliminaDati() {
   localStorage.clear();
   mostraDati();
}
</script>

</head>
<body onload="mostraDati()">
<form>
   <p>
   Nome:<input type="text" id="nome">
   Cognome:<input type="text" id="cognome">
   <input type="reset" value="Annulla" />
   </p>
   <p>
   <input type="button" onclick="salvaDati()" value="Salva" />
   <input type="button" onclick="mostraDati()" value="Mostra" />
   <input type="button" onclick="eliminaDati()" value="Elimina" />
   </p>
</form>
<p id="datiSalvati"></p>
</body>
</html>
```

All'apertura della pagina Web (evento **onload**), viene chiamata la funziona *mostraDati*.



Chiudendo e riaprendo la finestra (oppure ricaricando la pagina con il pulsante *Aggiorna* del browser), si può osservare che i dati rimangono memorizzati.

Se non ci sono dati memorizzati viene mostrata la stringa *null*.
Si può intercettare questa situazione e gestirla con una struttura *if*, modificando il codice della funzione *mostraDati*:

```
function mostraDati() {
   if(localStorage.getItem("nome")==null ||
         localStorage.getItem("cognome")==null){
      document.getElementById("datiSalvati").innerhtml = "nome o cognome
         non memorizzato";
   } else {
   var nome = localStorage.getItem("nome");
   var cognome = localStorage.getItem("cognome");
   document.getElementById("datiSalvati").innerhtml = nome + " " + cognome;
   }
}
```