

# Tecnologie Web

## Realizzazazione di un sito WEB

Hotel Supramonte :: Il vostro angolo di paradiso alpino

Docente: Ombretta Gaggi

Anno Accademico 2008/2009

I CSS per caso

Bernardinello Sandro IF/586374

Carlesso Enrico IF/586563

Trevisiol Michele IF/586289

# Indice

# Capitolo 1

## Introduzione

### 1.1 Idea del progetto

Abbiamo deciso di realizzare un sito per un ipotetico albergo cadorino. Questa scelta ci ha permesso di includere tutti gli elementi trattati nel corso partendo da layouting (xhtml, css) fino alla programmazione (perl, xml, javascript).

Per decidere la strutturazione del sito abbiamo effettuato una ricerca sul web ed abbiamo esaminato svariati siti della stessa categoria, per poter aver chiare quali potrebbero essere le esigenze di un eventuale committente.

Abbiamo quindi capito che particolare attenzione viene posta alle pagine divulgative, che devono essere contemporaneamente semplici ed accattivanti. Contemporaneamente però ci si aspetta che il sito offra delle interazioni che usualmente riguardano la possibilità di inviare delle prenotazioni e un semplice guestbook.

### 1.2 Pianificazione del lavoro

Dopo uno studio iniziale sono stati individuati i seguenti abilitati di sviluppo per dare forma al sito:

- Layouting (Ideazione grafica, *xhtml* statico e *css*);
- Strutturazione del sito in perl;
- Interfacciamento con dati *xml*;
- Codice *Javascript* per sezioni dinamiche e validazione dati;
- Valutazione del sito (rispetto standard *w3c*, accessibilità...);

Si è partiti dall'ideazione grafica per poter avere una base su cui lavorare. Sono quindi state realizzate le versioni statiche di tutte le pagine del sito, senza curare troppo in prima fase di stesura la pulizia del codice né tantomeno la validazione, preferendo fare un lavoro di rifinitura, una volta chiara l'architettura del sito.

Una volta definite le sezioni si è proceduto con la progettazione della struttura del sito, generando tutte le pagine con l'interprete perl.

Sono state scritte le pagine pesantemente basate su perl per le due sezioni con interfacciamento a dati xml del sito, le pagine di richiesta della prenotazione e il guestbook.

Una volta imbastita la struttura sono stati inseriti degli script in javascript, sia per dare qualche effetto gradevole, sia per la gestione di alcune pagine dinamiche o semi-dinamiche.

Come ultimo step si è portato il codice a rispettare gli standard **w3c** e sono state applicate le linee guida per l'accessibilità.

## Capitolo 2

# Struttura del sito

### 2.1 Introduzione

Il sito web in alcune circostanze è qualcosa di più di una semplice pagina informativa. Per un hotel, ad esempio, il sito web rappresenta una *presentazione* della propria attività, un modo di farsi conoscere, di farsi pubblicità e quindi di colpire o meno l'attenzione dell'utente.

Si è cercato quindi di rispettare una serie di caratteristiche ritenute idonee al progetto:

- SEMPLICITÀ: l'utente deve capire immediatamente la struttura del sito, non deve potersi perdere né ricevere troppe informazioni inutili;
- CHIAREZZA: deve essere chiara la struttura della pagina, partendo dal menu, in modo che l'utente possa controllare *in pochi click* le caratteristiche che cerca nell'hotel;
- COMPATTEZZA: si è preferito una struttura non in grado di *stupire* per gli effetti, ma piuttosto per la disposizione dei contenuti e per come questi vengono presentati;
- ACCESSIBILITÀ: la fruibilità del sito deve essere garantita a tutti gli utenti del web.

Poichè la struttura del sito prevede, per ogni pagina, una parte iniziale (*header*) ed una finale (*foot*) costanti, con il duplice scopo di semplificare lo sviluppo e di mantenere una perfetta coerenza nelle varie sezioni del sito, abbiamo deciso di utilizzare l'interprete perl per la generazione delle pagine, mantenendo una sola versione degli elementi condivisi.

### 2.2 Layout

**index.html** È una pagina di **presentazione** del sito dove, in un semplice riquadro, vengono messi in risalto i principali servizi forniti dall'hotel, sono presenti delle foto per dare un'idea visiva all'utente dell'hotel ed infine le informazioni di locazione e comunicazione con la struttura.

**index.cgi?section=\*** Le altre pagine "di navigazione" sono strutturate in modo molto semplice. Prevedono tre sezioni principali della pagina, la parte superiore (**home**) identifica il *nome* dell'hotel con il *riassunto* dei servizi. La parte centrale (**scr**) comprende il menu e il contenuto vero e proprio delle informazioni, rispettivamente chiamati **nav** e **corpo**. L'ultima parte è semplicemente una *barra informativa* che contiene le informazioni per un immediato contatto con l'hotel ed è identificata con il nome **foot**.

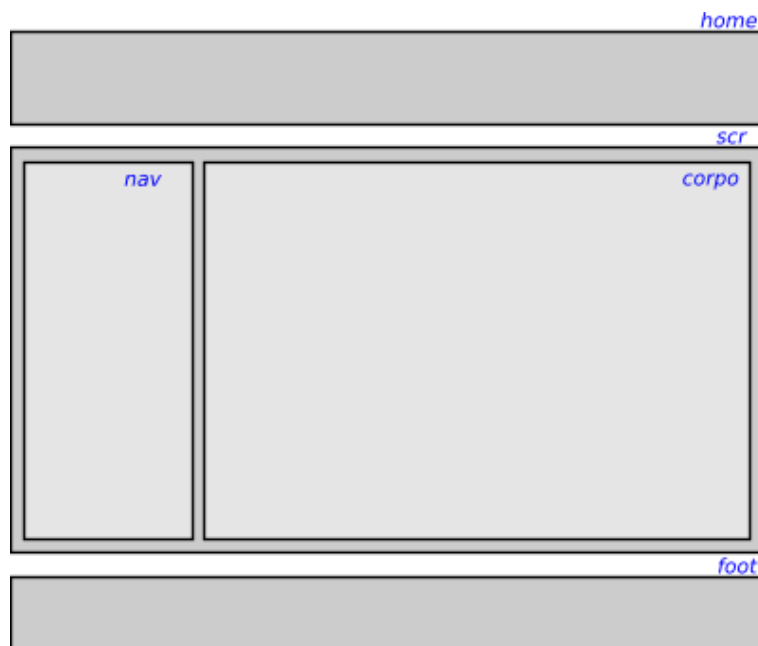


Figura 2.1: Rappresentazione struttura sito

Come già anticipato, la parte *variabile* è solamente quella centrale, precisamente solo il *menu* e il *corpo*. Il primo prevede sempre le stesse sezioni ma variano i colori dei pulsanti per segnalare la sezione corrente. Nel secondo invece variano ovviamente tutti i contenuti essendo il vero “corpo” del sito.

## 2.3 Utilizzo CSS

Sono presenti diversi fogli di stile a seconda della funzione richiesta, di seguito si descrivono brevemente:

**homescreen.css** è utilizzato solo dalla pagina di *presentazione* “index.html” e contiene solo le parti relative ad essa, come l’elenco dei servizi composto da immagini con descrizione a lato. Si è deciso di mantenere isolata questo foglio di stile proprio per il suo uso limitato a questa “particolare” pagina;

**screen.css** è il foglio di stile principale; tutte le pagine del sito “normali” seguono queste regole, dalla struttura di *home* e *foot*, alla disposizione delle immagini nelle varie pagine del *corpo*. Include alcuni semplici effetti e tratta anche sezioni “particolari” come PRENOTAZIONE e GUESTBOOK;

**print.css** è il foglio di stile chiamato dal browser quando l’utente richiede di stampare la pagina visualizzata. Si basa sul precedente “screen.css” eliminando alcune aree come *nav* e molte parti legate ad effetti o a tabelle non ritenute necessarie alla stampa. Inoltre prevede come colore principale dei caratteri il nero, vanno però esclusi alcuni titoli e gli “highlight” (per evidenziare numeri o parole ritenute importanti);

**calendar.css** contiene le definizioni di stile del calendario utilizzato per selezionare la data di arrivo nella sezione di prenotazione. Si riferisce esclusivamente agli elementi generati dal codice javascript.

### 2.3.1 Struttura home e scr

La parte superiore, come detto, prende il nome di *home* poichè la costante principale di ogni pagina, comprende infatti il titolo e un “riassunto” dei servizi offerti dall’hotel.

Interessante segnalare l’implementazione del titolo *Hotel Supramonte* che impiega una delle tecniche di **Image Replacement**. Questa prevede di utilizzare un’immagine come titolo della pagina senza perdere accessibilità, ovvero ponendo dietro all’immagine il titolo scritto in modo da renderlo leggibile agli screen reader e agli utenti che non possono visualizzare le immagini.

```
#home > h1 {
    position: relative;
    width: 570px;
    height: 68px;
    left: 5%;
    margin-top: 15px;
    margin-bottom: 5px;
    float: left;
}

#home > h1 span {
    position: absolute;
    top: 0;
    background-image: url(img/bordo/title_big_red.png);
    width: 570px;
    height: 68px;
}
```

La porzione di codice xhtml relativa a questo titolo si presenta così: `<h1><span></span>Hotel Supramonte</h1>`, includedo uno `<span>` vuoto al quale viene imposto come sfondo l’immagine relativa al titolo.

Questa tecnica presenta dei problemi di ridimensionamento poichè l’immagine ha delle dimensioni fisse ben specificate; per ovviare a questo e per una continua leggibilità delle pagine anche a basse risoluzioni, si è imposta una larghezza minima nel campo scr.

Lo sfondo della sezione “scr” è stato *scomposto* in tre parti: l’immagine sfocata principale è stata messa come background proprio in scr, mentre al menu e al corpo è stata imposta un’immagine ripetuta in alto per dare l’idea di una separazione tra la sezione “home” e “scr” stessa. Infatti, escludendo le immagini dalla pagina, si vede uno sfondo unico per tutte le sezioni senza che siano effettivamente separate, come si può vedere in (??).

### 2.3.2 Struttura menu e corpo

Il menu è semplicemente una lista di link implementata in perl in modo da distinguere la sezione corrente cambiando la classe dell’elemento nel menu. Per questo si utilizza una tecnica di layout chiamata *rollover*, dove si impiega un’unica immagine evidenziandone parti diverse a seconda dell’azione in corso. Nel nostro caso la 2.2 prevede lo sfondo da applicare normalmente per “simulare” un pulsante, quando è evidenziata dal mouse o quando è selezionata; mentre il nome delle sezioni è semplicemente centrato sul proprio sfondo permettendo sempre un’accessibilità costante e coerente con le motivazioni dell’impiego dell’Image Replacement.

Il corpo delle pagine prevede i contenuti del sito, contiene principalmente informazioni testuali ma anche diverse immagini. Queste sono state suddivise in quattro diverse categorie a seconda di come devono apparire nella pagina: *full*, *middle*, *map* e ovviamente quella di default che prevede la configurazione standard delle immagini. Semplicemente:

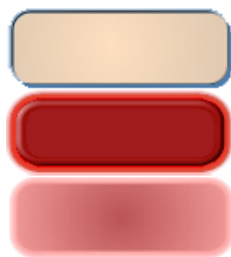


Figura 2.2: Immagine utilizzata per il rolling

**default** prevede di affiancare le immagini regolandone l'altezza in percentuale, si ottiene così una visione elegante, soprattutto se queste hanno le medesime dimensioni.

**full** utilizzata poche volte per le immagini larghe ma non tanto alte. Queste occupano quasi l'intera parte del corpo, ma hanno un significato più rilevante delle altre immagini. (es. foto del personale, immagine principale dell'albergo. . . )

**middle** è una configurazione più rilevante di quella di default ma non copre l'intera larghezza del corpo

**map** utilizzata solamente nella sezione *contatti* per la mappa presa da *google maps*, questa poi viene zoomata attraverso un effetto particolare ma l'immagine normale è definita da queste regole

Inoltre è stata introdotta una classe *virtualbox* che visualizza le immagini sfocate dimezzando l'opacità con cui vengono visualizzate dal browser, riportandola a livello normale quando il mouse vi si trova sopra.

Questa tecnica è utilizzata solo per le foto dell'hotel: per cadere in secondo piano rispetto al testo non “distraindo” così l'utente, ma anche per dare l'effetto opposto, ovvero “richiamando” la sua attenzione. Tuttavia queste poche righe di CSS non sono ancora divenute standard e quindi non superano la validazione; un'alternativa sarebbe quella di implementare lo stesso meccanismo in JavaScript, ma si otterrebbe un risultato più pesante se applicato ad un gran numero di immagini. Un'altra soluzione possibile sarebbe di applicare la stessa tecnica utilizzata nei pulsanti del menu: il *rolling*, ma bisognerebbe implementarla per ogni immagine raddoppiandone così il numero totale; questa quindi non rappresenta sicuramente una soluzione ottimale. A fronte di ciò si è quindi scelto di utilizzare comunque queste righe a discapito della validazione globale.



Figura 2.3: Visualizzazione foto normali o con offuscamento

Oltre alle regole relative alle immagini, la parte del foglio di stile del corpo riguarda anche i form della sezione *prenotazione* e la visualizzazione del *guestbook*. Si è cercato di implementare un CSS semplice ed intuitivo mantenendo le pagine chiare e pulite seppur includendo qualche effetto gradevole.

## 2.4 Perl - Generazione Pagine

Come descritto nelle sezioni precedenti, le parti in comune delle pagine ben si prestano ad essere gestite attraverso il perl.

La struttura tipica di una richiesta è la seguente

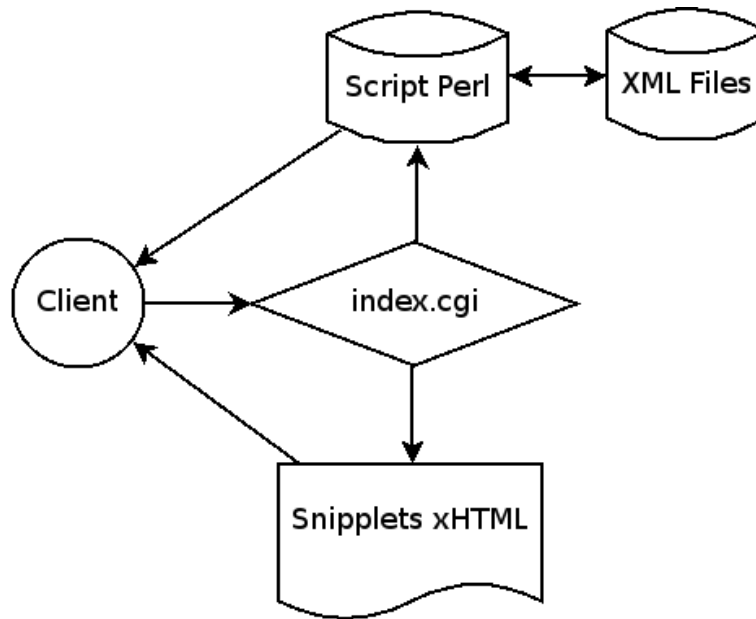


Figura 2.4: Rappresentazione di una richiesta

Il client effettua una chiamata ad `index.cgi` passando in GET la sezione richiesta, ad esempio `index.cgi?section=albergo`, sarà quindi compito dello script di index di capire che sezione è stata richiesta e conseguentemente di preparare i dati per la visualizzazione.

Vediamo nel dettaglio:

```
#!/usr/bin/env perl

require "utils.pl";
$moo = '<script type="text/javascript" src="../js/mootools-core.js"></script>
      <script type="text/javascript" src="../js/mootools-more.js"></script>';
$vbox = <<VBOX;
      <script type="text/javascript" src="../js/moord.js"></script>
      <script type="text/javascript">
//
window.addEvent('load', function() {
    var virtual = new Virtual.Box({</pre></div>
```



```

        enable: {
            arrows: true,
            closeButton: true
        },
        effect: 'open',
        leftArrowText: '',
        rightArrowText: '',
        closeButtonText: '',
        captionOpacity: 0.6,
    });
});
//]]>
</script>
VBOX
%this = parse_add($ENV{'QUERY_STRING'});
%sections = (
    'home' => {
        'title' => 'Hotel Supramonte :: Home',
        'page' => '../snippets/home.htm',
        'script' => '',
        'type' => 'html',
        'css' => ''
    },
    'chisiamo' => {
        'title' => 'Hotel Supramonte :: Chi Siamo',
        'page' => '../snippets/chisiamo.htm',
        'script' => '',
        'type' => 'html',
        'css' => ''
    },
    'ristorante' => {
        'title' => 'Hotel Supramonte :: Ristorante',
        'page' => '../snippets/ristorante.htm',
        'script' => $moo.$vbox,
        'type' => 'html',
        'css' => ''
    },
    'albergo' => {
        'title' => 'Hotel Supramonte :: Albergo',
        'page' => '../snippets/albergo.htm',
        'script' => $moo.$vbox,
        'type' => 'html',
        'css' => ''
    },
    'prenotazioni' => {
        'title' => 'Hotel Supramonte :: Prenotazioni',
        'page' => 'prenotazio.pl',
        'script' => $moo.'<script type="text/javascript" src="../js/calendar.js">
            </script>'. "\n". '<script type="text/javascript" src="../js/prenotazioni.js"></script>',
    },

```

```

        'type' => 'perl',
        'css' => '<link rel="stylesheet" media="screen" href="../calendar.css" type="text/css" />'
    },
    'servizi' => {
        'title' => 'Hotel Supramonte :: Servizi',
        'page' => '../snippets/servizi.htm',
        'script' => $moo.$vbox,
        'type' => 'html',
        'css' => ''
    },
    'guestbook' => {
        'title' => 'Hotel Supramonte :: GuestBook',
        'page' => 'guestbook.pl',
        'script' => $moo.'<script type="text/javascript">document.addEvent("domready", function() {
            if($("#last_insert")) $("#last_insert").highlight();</script>',
        'type' => 'perl',
        'css' => ''
    },
    'contatti' => {
        'title' => 'Hotel Supramonte :: Contatti',
        'page' => '../snippets/contatti.htm',
        'script' => '',
        'type' => 'html',
        'css' => ''
    },
    'pren_result' => {
        'title' => 'Hotel Supramonte :: Richiesta inviata',
        'page' => 'pren_result.pl',
        'script' => '<script type="text/javascript" src="../js/calendar.js"></script>'.
            "\n".<script type="text/javascript" src="../js/prenotazioni.js"></script>',
        'type' => 'perl',
        'css' => ''
    }
};

```

```

print "Content-type: text/html\n\n";
$head = plain_return('../snippets/header.htm');
$head =~ s/###TITLE###/$sections{$this{'section'}}->{'title'}/;
$head =~ s/###JAVASCRIPT###/$sections{$this{'section'}}->{'script'}/;
$head =~ s/###CSS###/$sections{$this{'section'}}->{'css'}/;
$head =~ s/###$this{'section'}###/active/;
$head =~ s/###(.*)###/default/g;
print $head;
for ($sections{$this{'section'}}->{'type'}) {
    if (/html/) {
        plain_include($sections{$this{'section'}}->{'page'});
    }
};

```

```

    if(/perl/) {
        require $sections{$this{'section'}}->{'page'};
    };
}
plain_include('../snippets/footer.htm');

```

Come si può vedere viene mantenuta una struttura del sito mediante un doppio Hash, dove vengono descritte le informazioni necessarie per la generazione della pagina.

All'inizio viene incluso un file che contiene tre semplici funzioni di appoggio:

**parse\_add** Una semplice funzione che restituisce un hash rappresentante l'associazione chiave-valore presente nell'url: all'indirizzo `index.cgi?section=chisiamo` quindi verrà chiamato *parse\_add* con i parametri `section=chisiamo` che restituirà un Hash così strutturato: `{'section' => 'chisiamo'}` che viene utilizzato nella generazione della pagina;

**plain\_include** Un'altro triviale metodo che ha il semplice compito di stampare riga per riga un certo file (con altre funzioni native del perl il file verrebbe interpretato come listato perl, contrariamente alle nostre necessità);

**plain\_return** Questa metodo invece restituisce in una sola stringa tutto il contenuto di un file.

I files (`'../snippets/header.htm'`) e (`'../snippets/footer.htm'`) contengono le porzioni di codice xhtml che compongono l'home, menu e foot di figura (2.1). A differenza del footer, che non contiene alcuna parte modificabile da pagina a pagina, l'header contiene delle informazioni specifiche per la pagina.

Abbiamo quindi inserito nel codice xhtml degli snippets delle stringhe fittizie in posizioni chiave della pagina per poter successivamente sostituire queste stringhe con le informazioni necessarie. Si è fatto così per il titolo, per gli script javascript da includere e per evidenziare la sezione attiva del menu, ricavando dall'hash di struttura le informazioni adeguate. Da qui la necessità del metodo **plain\_return**. Con questa tecnica è stato possibile includere script e/o css aggiuntivi solo dove necessario, evitando di istanziare il visualizzatore javascript nelle pagine sprovviste di immagini.

Viene poi discriminato il "tipo" della pagina corrente, che può essere **html** o **perl**. Nel caso di html la pagina viene direttamente stampata, nel caso di listato per la pagina viene inclusa con la chiamata **require**, che per l'appunto interpreta il file a parametro come codice perl.

Sarà quindi cura di questi script generare l'output html necessario.

### 2.4.1 Prenotazioni

Il codice html della sezione **prenotazioni** viene generata da uno script perl che gestisce i dati dei form in modo dinamico, come ad esempio il campo data che viene posto al giorno corrente.

I dati inseriti dall'utente vengono spediti utilizzando il metodo POST; vengono quindi interpretati dallo script che effettua i controlli di correttezza sui campi e che va poi a salvare il tutto sui file xml corrispondenti.

In caso di compilazione non corretta o incompleta del form, vengono evidenziati i campi errati in javascript; altrimenti viene invece generata un'altra pagina con un riepilogo dei dati inseriti per la prenotazione.

Va menzionata una nota sull'accessibilità, ovvero che in caso di un browser con javascript disabilitati, il controllo viene comunque effettuato in perl mantenendo tutti i form già compilati correttamente.

## 2.5 XML

La scelta dell'XML come "formato" di archiviazione dei dati nasce dal fatto che permette una definizione semplice, chiara e diretta della struttura dati, facilmente implementabile e validabile. La gestione dei dati,

dall'acquisizione dai form al salvataggio nel file xml, è stata effettuata utilizzando Perl come linguaggio di scripting. Il binomio Perl-XML presenta notevoli vantaggi grazie alle svariate funzioni di supporto per l'I/O fornite nella libreria **Lib::XML**, con le quali abbiamo anche validato i file in base agli XML-Schema definiti precedentemente.

### 2.5.1 Prenotazioni

Questa sezione permette all'utente di mandare una richiesta di prenotazione allo staff dell'hotel (noi!) con i suoi dati personali e i dati relativi all'ipotetico soggiorno presso l'albergo.

I dati sono:

**Dati personali:** nome, cognome, indirizzo, numero di telefono ed email.

**Dati prenotazione:** data di arrivo, numero di notti di soggiorno, numero di persone, numero di camere da prenotare per ogni tipologia di camera (singola, doppia o quadrupla), la tipologia del soggiorno ed eventuali richieste particolari

La struttura dati è definita dagli schemi XML seguenti:

```
<?xml version="1.0"?>
<ht:schema xmlns:ht="http://www.w3.org/2001/XMLSchema">
  <ht:element name="clienti">
    <ht:complexType>
      <ht:sequence>
        <ht:element name="cliente" minOccurs="0" maxOccurs="unbounded">
          <ht:complexType>
            <ht:attribute name="email" type="ht:string"/>
            <ht:sequence>
              <ht:element name="nome" type="ht:string"/>
              <ht:element name="cognome" type="ht:string"/>
              <ht:element name="indirizzo" type="ht:string"/>
              <ht:element name="tel" type="ht:string"/>
            </ht:sequence>
          </ht:complexType>
        </ht:element>
      </ht:sequence>
    </ht:complexType>
  </ht:element>
</ht:schema>

<?xml version="1.0"?>
<ht:schema xmlns:ht="http://www.w3.org/2001/XMLSchema">
  <ht:element name="prenotazioni">
    <ht:complexType>
      <ht:sequence>
        <ht:element name="prenotazione" minOccurs="0" maxOccurs="unbounded">
          <ht:complexType>
            <ht:sequence>
              <ht:element name="cliente" type="ht:string"/>
              <ht:element name="data_arrivo" type="ht:string"/>
              <ht:element name="arrivo_giorno" type="ht:string"/>
            </ht:sequence>
          </ht:complexType>
        </ht:element>
      </ht:sequence>
    </ht:complexType>
  </ht:element>
</ht:schema>
```

```

<ht:element name="arrivo_mese" type="ht:string"/>
<ht:element name="arrivo_anno" type="ht:string"/>
<ht:element name="notti" type="ht:string"/>
<ht:element name="singole" type="ht:string"/>
<ht:element name="doppie" type="ht:string"/>
<ht:element name="quadruple" type="ht:string"/>
<ht:element name="pensione" type="ht:string"/>
<ht:element name="persone" type="ht:string"/>
<ht:element name="richieste" type="ht:string"/>
</ht:sequence>
</ht:complexType>
</ht:element>
</ht:sequence>
</ht:complexType>
</ht:element>
</ht:schema>

```

Sono presenti due elementi principali, ovvero: `<prenotazione>` e `<cliente>`. Il primo viene creato ogni qualvolta viene effettuata una richiesta in modo corretto e ha sempre tutti i sottoelementi non vuoti, a meno delle richieste particolari che sono appunto opzionali. Per quanto riguarda le informazioni legate all'utente che effettua la richiesta di prenotazione, viene salvato solo il campo email che funge da chiave nella relazione prenotazione-cliente.

Il file contenente i dati relativi a tutti i clienti contiene elementi di tipo `<cliente>` la cui univocità è basata sul campo *email* che si è deciso di salvare come attributo. Questa scelta dipende dal fatto che, al momento della ricerca all'interno del file, non si devono "scorrere" tutti gli elementi figli nell'equivalente albero rappresentante la struttura del file stesso, garantendo così una velocità di ricerca maggiore.

## 2.5.2 Guestbook

La sezione guestbook contiene un form con il quale l'utente inserisce un messaggio nel libro visite del sito.

La struttura del file *guestbook.xml* è costituita da un elemento principale, `<guestbook>`, e da tutti gli elementi figli chiamati `<item>`. Ogni `<item>` rappresenta un messaggio lasciato e contiene i campi `<mittente>`, `<contenuto>` e `<data>`. Inoltre ogni `<item>` ha anche un attributo *ID* univoco che serve per la visualizzazione dei messaggi nell'ordine temporale corretto.

Durante i vari test di utilizzo è stato riscontrato un problema dovuto all'inserimento di caratteri non ascii all'interno dei messaggi inseriti nel guestbook come, ad esempio, le lettere accentate; queste non venivano riconosciute come carattere UTF-8 e di conseguenza visualizzate in modo scorretto. L'inconveniente è stato risolto ponendo un controllo, prima del salvataggio su file, che sostituisce i caratteri speciali con la stringa corrispondente nella codifica UTF-8 e la salva nel file xml.

## 2.6 Integrazione con JavaScript - Mootools

Abbiamo ritenuto adeguato e proficuo integrare nel nostro sito degli script in Javascript, che descriveremo in seguito. Per semplificare il lavoro, ma soprattutto per aver la certezza della corretta interpretazione da parte di tutti i browser di ogni sistema operativo, abbiamo deciso di lavorare utilizzando il framework mootools ([www.mootools.net](http://www.mootools.net)).

Come presentato dagli stessi autori:

Mootools è un framework JavaScript compatto, modulare e Object-Oriented, disegnato per sviluppatori Javascript esperti. Permette di scrivere codice potente, flessibile e cross-browser con le sue API eleganti, ben documentate e coerenti.

È un framework rilasciato sotto la licenza Open Source MIT. Come nota di cronaca va segnalato che questo framework è quello utilizzato dallo stesso w3c nel proprio validatore ([validator.w3.org](http://validator.w3.org)), oltre che dal CMS joomla, il sito dell'ubuntu e svariati altri nomi rilevanti. Ulteriore informazione degna di nota, il "project leader" è italiano, Valerio Proietti.

Per quanto riguarda il nostro sito siamo ricorsi a JavaScript con due scopi differenti.

Abbiamo voluto inserire un tool che genera un semplice album ogni volta che un utente clicca su una foto nelle pagine di "vetrina", permettendogli di scorrere tra le foto della pagina:

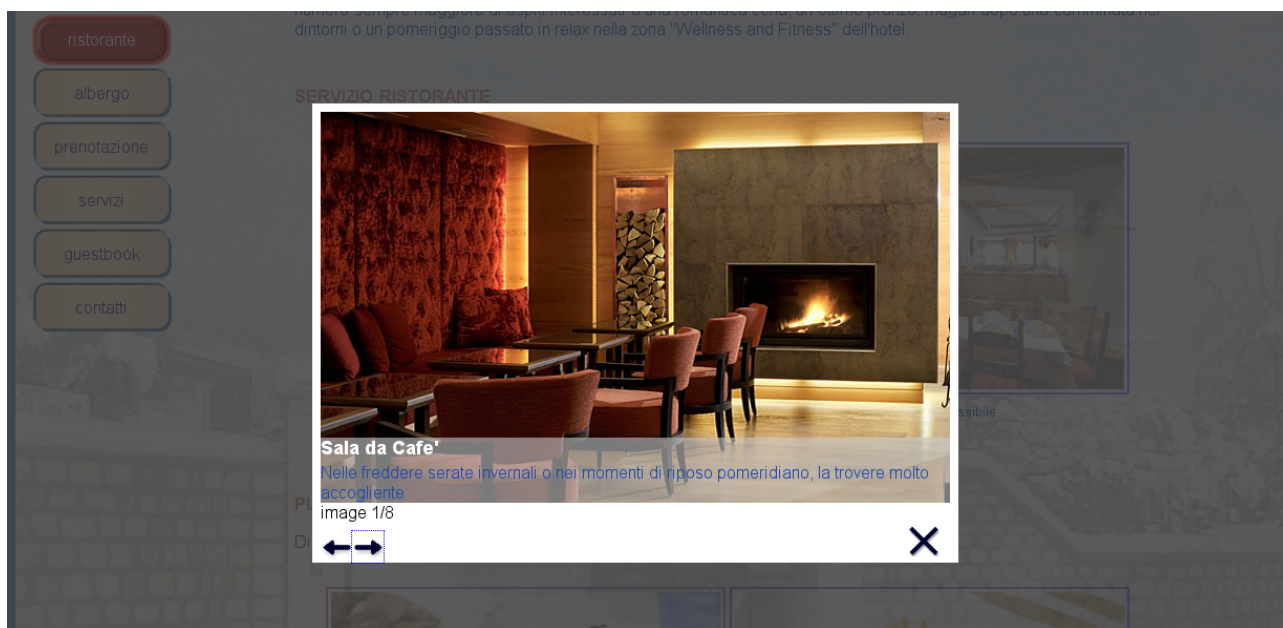


Figura 2.5: Screenshot di esempio delle foto

Meno scenica ma più utile è stato l'utilizzo di JavaScript nella pagina contenente il form di prenotazione, nella quale si è voluto inserire dei controlli per la corretta compilazione del form per poter effettuare una prima validazione dei dati, sia per valutarne la completezza sia la correttezza.

Al momento dell'invio del form viene quindi invocato il nostro script che effettua un controllo dei dati e in caso di non corretta compilazione ferma l'invio, evidenzia l'errore nella pagina e riposiziona il browser al primo errore trovato.

Sfruttando la potenza di Mootools, il compito diventa relativamente semplice, soprattutto non dovendosi preoccupare della compatibilità tra i browser.

Per illustrare la potenza del framework utilizzato, presentiamo un paio di stralci dallo script che abbiamo scritto.

```
$( 'pren_form' ).addEvent( 'submit', function( e ) {  
    e.stop();  
    ...  
});
```

Già da queste prime due righe si intuisce la potenza di Mootools. Il selettore `$('<id>')` restituisce l'elemento della pagina contraddistinto da quello specifico id. Si noti che esiste anche la versione `$$('<css-selector>')` che permette di utilizzare un qualsiasi selettore css, restituendo un array contenente tutti gli elementi.

La funzione `addEvent(event, function())` permette di “bindare” al raising di un evento su uno specifico elemento, una funzione predefinita. Nel nostro caso abbiamo quindi agganciato al form di prenotazione (contraddistinto per l'appunto dall'id `pren_form`) la funzione che controlla i campi, la quale viene invocata quando l'utente clicca sul tasto Invia.

La prima cosa da fare è, ovviamente, fermare l'invio del form, che viene eseguito mediante l'istruzione `e.stop()` dove `e` è il parametro passato alla funzione.

Vengono poi controllati tutti i campi ritenuti obbligatori (nome, cognome, indirizzo, telefono, email), nel caso risultino vuoti o non validi viene inserita una stringa al di sopra del campo, notificando l'errore:

```
if(!$('nome').value) {
    if(!inval.contains($('nome'))){
        nome_error = new Element('div', {
            'id': 'nome_error',
            'html': 'Inserire il nome'
        });
        nome_error.inject($('nome').getParent(), 'top');
        $('nome').getParent().addClass('error');
    }
}
else {
    if(inval.contains($('nome'))){
        inval.splice(inval.indexOf($('nome')), 1)
        $('nome_error').dispose();
        $('nome').getParent().removeClass('error');
    }
}
```

Viene controllato il contenuto dell'input, se non valido viene generato un div che viene inserito tramite la funzione `inject()`, viene inoltre aggiunta la classe 'error' all'elemento che contiene l'input invalido.

In caso contrario viene verificato se precedentemente il campo fosse stato invalido, nel qual caso rimuove il div con la comunicazione di errore e rimuove la classe dal padre. Possiamo vederlo in azione in figura (??).


The figure consists of two side-by-side screenshots of a web form titled "DATI PERSONALI:". The form has five input fields: "Nome:", "Cognome:", "Indirizzo:", "Telefono:", and "E-mail:". In the left screenshot, all fields are filled out: "Nome:" is "Paolino", "Cognome:" is "Paperino", "Indirizzo:" is "Via Roma, 23", "Telefono:" is "04912121212", and "E-mail:" is "paperino@math.unipd.it". In the right screenshot, the "E-mail:" field is empty and highlighted with a red border, indicating an error. The other fields are still filled out.

Figura 2.6: Form correttamente compilato e form incompleto

Sempre nella pagina di invio form è stato inserito un calendario a scomparsa per l'inserimento della data (??).

Viene attivato quando l'input della data genera l'evento focus o quando si clicca sull'icona a destra del form.

**DATI PRENOTAZIONE:**

Data di arrivo (gg/mm/aaaa)  

Notti:  ▼

Persone:  ▼

Camere singole:  ▼

Camere doppie:  ▼

Camere quaduple:  ▼

Tipologia di soggiorno:

Richieste particolari:

Marzo, 2009 ▶

L	M	M	G	V	S	D
23	24	25	26	27	28	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

Figura 2.7: Calendario per la selezione data

Teniamo a precisare che tutte queste add-on in javascript non compromettono assolutamente l'usabilità del sito, vogliono semmai essere un aiuto. Infatti sono superflue, ovvero disabilitando l'interprete javascript del browser, al posto dell'input per la data vengono visualizzati tre select per la scelta della data, e la validazione dei dati viene fatta in perl, che in caso di incompleta compilazione lo notifica all'utente e gli permette di completare i campi mancanti.



## Capitolo 3

# Accessibilità

### 3.1 Introduzione

L'**accessibilità** è la capacità di una risorsa, sia essa un dispositivo o semplicemente il contenuto informativo di una pagina web, di essere fruibile con facilità da tutte le categorie di utenti che possono accedervi.

Un sito web pubblico, come quello sviluppato per questo progetto, ha come obiettivo primario la divulgazione delle informazioni contenute in esso; queste informazioni devono poter essere raggiunte da tutta l'utenza che può rappresentare, nel nostro caso, una potenziale clientela per l'hotel.

Per garantire che questo requisito sia soddisfatto, è bene sviluppare il sito seguendo alcune linee guida:

- rispettare gli standard del *W3C*;
- separare il contenuto dal layout;
- garantire la visualizzazione chiara del contenuto su supporti diversi;
- utilizzare sempre un testo alternativo per i contenuti multimediali;
- utilizzare *tabindex* e *accesskey*;
- fornire sempre i label per gli input...

### 3.2 Implementazioni

Durante lo sviluppo del sito sono stati seguiti alcuni importanti accorgimenti per garantire l'accessibilità; in particolare per quanto riguarda le immagini e le pagine in cui è presente un'interazione attiva dell'utente, ovvero dove sono presenti dei form.

#### 3.2.1 Menu

Tutti i link del menu sono stati dotati di adeguato "accesskey", nonostante sia un delicato argomento sul quale esistono svariate opinioni contrastanti e soprattutto manca un'uniformità tra i client web.

Il sito [www.webaim.org](http://www.webaim.org), che si occupa di accessibilità dà delle linee guida per gli accesskey, consigliando di usare valori numerici del tipo: 1 - Home Page, 5 - FAQ...

Seguendo questi principi abbiamo assegnato ad ogni link del menu un accesskey numerico da 1 a 8 e fornendo il *tabindex* a tutti. Per comodità di navigazione tra le pagine, con particolare attenzione a quella di prenotazione,

i tabindex sono stati numerati a partire da 90, così da poter far fluire i tab prima per tutti i vari elementi della pagina.

Ulteriore accorgimento per la gestione degli accesskey consiste nel segnalare l'accesskey all'utente attraverso qualche modalità. Webaim spiega alcuni modi per evidenziare la presenza di accesskeys. Abbiamo deciso di implementare la quinta modalità da loro suggerita:

```
# Use more sophisticated CSS and/or browser detection approaches that expose the accesskey
shortcuts when the elements receive focus or when the mouse hovers over them.

* Advantages: Does not interfere with visual layout; if both CSS and browser detection
  are used, then users can be notified of the exact keystroke combination necessary.
* Disadvantages: Requires working knowledge of advanced CSS and/or browser detection
  scripts.
```

Con perl abbiamo quindi individuato l'user-agent del browser che effettua la richiesta per poter capire quale fosse lo shortcut usato dal client in questione per accedere agli access-key (che variano da browser a browser e da sistema operativo a sistema operativo), abbiamo aggiunto ad ogni link del menu uno `<span>` contenente shortcut ed accesskey, normalmente resi invisibili tramite css (con `display:none` in modo che vengano saltati anche dai reader) e resi visibili solo se attivati con questa definizione css:

```
#nav li a:hover span.ak, #nav li a:focus span.ak {
  display: block;
}
```

Webaim fornisce anche una toolbar per firefox che consente di effettuare una valutazione sommaria sulla validità della pagina, controllando ad esempio che tutte le immagini siano dotate di testo alternativo, che siano sempre presenti i label per gli input, etc. È stato uno strumento utile soprattutto per correggere delle sviste che altrimenti sarebbero potute passare inosservate.

### 3.2.2 Immagini

Delle immagini sia intese come background sia come normali foto, si è già parlato nel capitolo relativo alla struttura con i riferimenti alle regole nei fogli di stile. Vediamo subito il risultato delle tecniche di *Image Replacement* e di come sono state sfruttate le immagini di sfondo per creare un layout gradevole senza danneggiare l'accessibilità.

L'immagine a sinistra rappresenta il sito visibile in una "normale" navigazione, ovvero intesa senza avere inaccessibilità di alcun tipo. La seconda rappresenta la stessa pagina privata di qualsiasi immagine, anche quelle di sfondo; si nota subito che la struttura viene mantenuta correttamente riportando un'ottima leggibilità. Questo è un vantaggio anche per gli screen reader che, non potendo distinguere le immagini, riescono comunque a "leggere" il testo senza errori: come il titolo o le voci del menu. Vengono "persi" solo i servizi in alto a destra che tuttavia sono rappresentati da una semplice immagine e non sono fondamentali nel contesto della pagina. Da notare che l'utente che fa uso di screen reader ha pieno accesso a tutte le informazioni (anche al numero di stelle dell'hotel già presente nella pagina *index.html* con la tecnica di IR).

Nel corpo delle pagine si trovano spesso immagini dell'hotel e dei servizi forniti, queste una volta cliccate vengono ingrandite tramite JavaScript con cui è inoltre possibile sfogliarle direttamente. Nel caso questi non siano abilitati, le immagini vengono visualizzate normalmente, cioè aperte a dimensione intera in una nuova pagina.

La sezione *contatti* contiene la cartina dell'hotel, questa è trattata in modo completamente diverso, effettua uno zoom quando il mouse si sovrappone alla mappa, utilizzando lo *z-index* nel CSS. Per avere sempre la



Figura 3.1: Confronto pagina con e senza immagini di sfondo

possibilità di ingrandire l'immagine anche quando questa caratteristica non viene supportata dai browser, è stato aggiunto un link direttamente sotto la mappa.

### 3.2.3 Form

L'inserimento di dati da parte dell'utente può avvenire in due diverse sezioni del sito: *Guestbook* e *Prenotazioni*.

Come in tutte le pagine del sito, anche qui, per garantire la completa accessibilità sono stati specificati i **tabindex** di tutti i campi di input ed è stato implementato il controllo dei campi obbligatori non correttamente compilati o lasciati vuoti.

Una particolare attenzione è stata utilizzata per quanto riguarda l'inserimento della data di arrivo; infatti si è cercato di garantire allo stesso tempo sia una maggiore facilità d'uso, sia l'accessibilità a tutti gli utenti. Per velocizzare e semplificare l'inserimento della data si è usato un componente javascript che fornisce un'interfaccia grafica di tipo calendario (??). Anche qui si è prestata particolare attenzione all'accessibilità, avendo cura di impostare i **tabindex** per tutti i giorni abilitati, modificando lo script in maniera che i giorni abbiano tutti **tabindex** di valore  $10 + i$  dove  $i$  è il numero del giorno corrente.

Curando appositamente gli altri **tabindex** presenti nella pagina, si può quindi scorrere attraverso i link mediante il tasto tab come schematizzato in figura:

È infine bene osservare che in caso un utente acceda alla sezione del sito utilizzando un browser con javascript disabilitati, l'accessibilità ai campi rimane comunque mantenuta dal codice generato dallo script perl, il quale si occupa anche di eseguire il controllo sui campi se non effettuato dal javascript apposito.

**DATI PERSONALI:**

Nome: **1**

Cognome: **2**

Indirizzo: **3**

Telefono: **4**

E-mail: **5**

I dati forniti verranno utilizzati soltanto al fine di elaborare la richiesta e non inoltrati a terzi (secondo le leggi n. 665/665 "Privacy": Ai sensi dell'articolo 10 e dell'articolo 13 della legge n. 675 del 31.12.1996 e successive modifiche, concernenti il trattamento dei dati personali). Ci distanziamo da qualsiasi pubblicità indesiderata (spam) e ci impegnamo ad evitare l'invio di e-mail non desiderate ed a limitarle solo per casi particolari.

**DATI PRENOTAZIONE:**

**6**  

Notti: **8**

Persone: **9**

Camere singole: **10**

Camere doppie: **11**

Camere quaduple: **12**

Tipologia di soggiorno **13** Solo pernottamento

Richieste particolari: **14**

Marzo, 2009

L	M	M	G	V	S	D
23	24	25	26	27	28	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

Figura 3.2: Rappresentazione percorso tra i link tramite tab