Le basi del web

Applicazioni web

Michele Tomaiuolo

tomamic@ce.unipr.it

http://www.ce.unipr.it/people/tomamic

Sommario

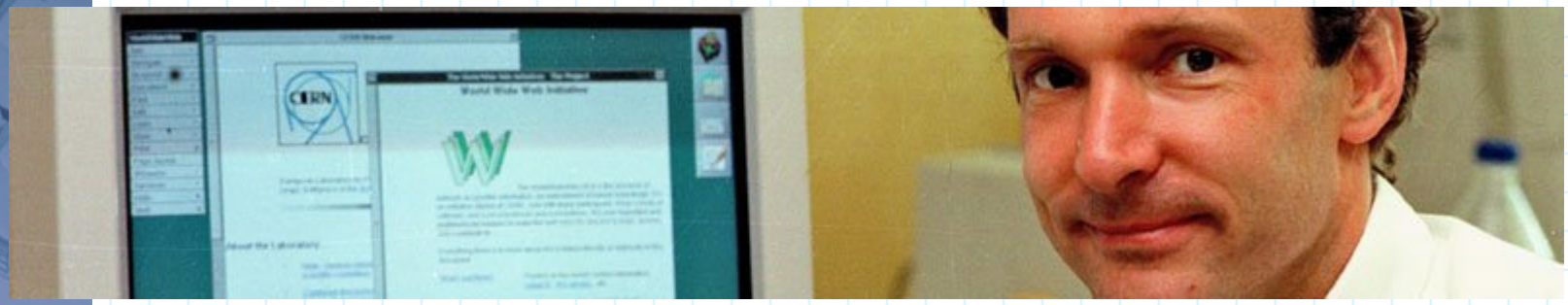
- Html: elementi principali ed esempi; collegamenti; liste e tabelle
- Form: azione e metodo; coppie nome/valore; campi di input
- Http: metodi, intestazioni e codici di stato; parametri e cookie
- Php: per pagine web attive

World Wide Web

- Sistema per la condivisione di informazioni ipertestuali
- Uno dei modi più diffusi di utilizzare la rete Internet, assieme alla posta elettronica
- Permette agli utenti di Internet di pubblicare e accedere a documenti HTML via HTTP
- Si basa su due programmi
  - Web server
  - Web client (browser)

Cronistoria del Web

- 1990: World Wide Web (HTML+HTTP)
  - Tim Berners-Lee, CERN, x doc scientifici
- 1993: Mosaic, primo browser grafico
  - Marc Andreessen
- 1994: Netscape Navigator
- 1995: Internet Explorer



HyperText Markup Language

- HTML permette di:
  - Pubblicare documenti strutturati con titoli, tabelle, foto ed elementi multimediali
  - Navigare facilmente tra le informazioni seguendo collegamenti ipertestuali
  - Presentare moduli per condurre interazioni con servizi remoti, al fine di ricercare informazioni, fare prenotazioni e acquisti
- Standardizzato dal W3C

http://www.w3.org/html/

Tag ed elementi

- HTML dichiara tipi di elementi
  - Per strutturare i documenti (paragrafi, liste, collegamenti ipertestuali, immagini ecc.)
- Tipo di elemento descritto da tre parti: un tag di apertura, un contenuto e un tag di chiusura
  - `Bla bla, <b>in grassetto.</b>, normale.`
- Molti tag permettono la definizione di attributi
  - `<a href="http://www.unipr.it">UniPR</a>`
- Tag semplici non hanno un contenuto
  - `<img src="blueribbon.gif" />`

Anatomia di una pagina

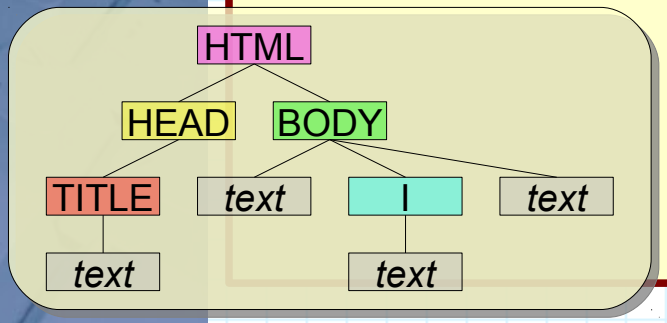
```
<html>
<head>
  <title>Hello, world</title>
</head>

<body>
  Hello, <i>world</i>!
</body>
</html>
```

Tipo di doc: racchiude l'intero file

Intestazione: informazioni descrittive

Titolo: deve essere nella intestazione

Corpo: parte principale, contenuto del doc

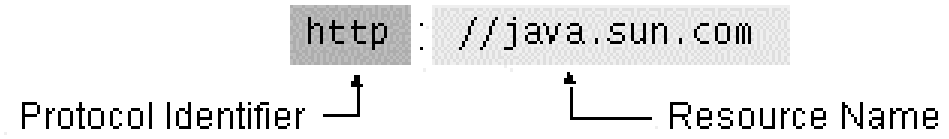


Tag di formattazione testo

```
<p>Questo è un paragrafo.</p>

<p>Prima linea.<br />A-capo ma stesso paragrafo.</p>

<p>Testo <strong>in grassetto</strong>, e poi
<em>in corsivo</em>.</p>

<h1>Il titolo più grande</h1>
...
<h6>Il titolo più piccolo</h6>

<pre>Testo preformattato: usato un font a larghezza
fissa; spazi e a-capo preservati</pre>

<!-- Commento HTML -->
```

Uniform Resource Locator

- Una URL è un riferimento per una risorsa



- Il nome della risorsa dipende interamente dal protocollo. Per HTTP include:
  - Nome dell'host su cui risiede la risorsa
  - Numero di porta cui collegarsi (default = 80)
  - Percorso della risorsa sulla macchina
  - Frammento: riferimento ad una sezione con id univoco all'interno della risorsa

http://www.ietf.org:80/rfc/rfc2732.txt

Collegamenti e citazioni

```
<p id="par1">Il primo paragrafo. Vai su
<a href="http://www.unipr.it/index.html#news">
UniPR, riquadro News</a>.</p>

<p id="par2">Il secondo paragrafo. Torna al
<a href="#par1">primo paragrafo.</a></p>

<!-- si può attribuire un id ad ogni elemento, ma
dev'essere univoco nella pagina -->

<blockquote cite="http://www.faqs.org/">
  <p>The BLOCKQUOTE element contains text quoted
    from another source.</p>
  <address><a href="http://www.faqs.org/">
    Da faqs.org</a></address>
</blockquote>
```

Liste

```
<ul>
    <li>First item</li>
    <li>Second item</li>
    <li>Third item</li>
</ul>

<ol>
    <li>First item</li>
    <li>Second item</li>
    <li>Third item</li>
</ol>
```

- First item
- Second item
- Third item

1. First item
2. Second item
3. Third item

Liste di definizione

```
<dl>
  <dt>First term</dt>
  <dd>Description of first term.</dd>
  <dt>Second term</dt>
  <dt>Third term</dt>
  <dd>Description of second and third term.</dd>
  <dd>Alternate description of second and third
      term.</dd>
</dl>
```

First term
    Definition of first term.
Second term
Third term
    Definition of second and third term.
    Alternate definition of second and third term.

Tabelle

- Le tabelle HTML servono per incasellare dati (e non per gestire il layout dell'intera pagina!)
- Sono marcate con `<table>`
- `<table>` contiene righe di celle, marcate con `<tr>` (dall'alto verso il basso ↓)
- Ogni `<tr>` contiene celle di dati, marcate con `<td>` (da sinistra a destra →)
- Le celle di intestazione sono marcate con `<th>`

Semplice tabella

```
<table>
    <tr>
        <td>northwest</td>
        <td>northeast</td>
    </tr>
    <tr>
        <td>southwest</td>
        <td>southeast</td>
    </tr>
</table>
```

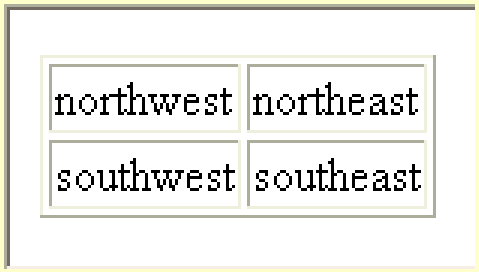


Tabella con celle unite

```
<table>
<caption><i>A test table with
merged cells</i></caption>
<tr>
  <th rowspan="2"></th>
  <th colspan="2">Average</th>
  <th rowspan="2">Red eyes</th>
</tr>
<tr>
  <th>height</th>
  <th>weight</th>
</tr>
<tr><th>Males</th>
  <td>1.9</td><td>0.003</td><td>40%</td></tr>
<tr><th>Females</th>
  <td>1.7</td><td>0.002</td><td>43%</td></tr>
</table>
```

A test table with merged cells

	Average		Red eyes
	height	weight	
Males	1.9	0.003	40%
Females	1.7	0.002	43%

Tag per computer

- Codice inline: `<code>`
- Blocchi di codice: `<pre><code>`
- Input da tastiera: `<kbd>`
- Variabili e costanti: `<var>`
- Output inline: `<samp>`
- Catture di schermate testuali (blocchi di output) `<pre><samp>`

Tag generici: div, span

- Aggiungere struttura e semantica ai documenti
  - `span` raggruppa contenuto inline
  - `div` raggruppa contenuto di livello blocco
  - Spesso assegnati attributi `id` e `class`
- Non impongono nessun altro vincolo di presentazione al contenuto
  - Gli autori possono usare questi elementi assieme a fogli di stile per adattare i documenti html ai loro bisogni e gusti

Id vs. class

- Mentre uno stesso valore di `class` può essere attribuito a molti elementi su una pagina...
- `id` deve essere unico all'interno del documento!
- Non si può applicare uno stesso valore di `id` a più elementi
- Si possono assegnare più classi ad un solo elemento, separate da spazio. Es.

```
<p class="news gossip">Bla bla.</p>
```

Markup semantico

- Documenti privi di markup di presentazione
- Definire un vocabolario di classi semantiche...

  da assegnare agli elementi con attributo `class`

  la cui presentazione può essere specificata in fogli di stile validi per tutto il sito
- Indicazioni di mozilla.org
  http://www.mozilla.org/contribute/writing/markup
- Accessibilità dei contenuti web
  http://www.w3.org/TR/WCAG10-HTML-TECHS/

Html 5

- Nuovi elementi di struttura
  - `header`, `nav`, `article`, `aside`, `footer`
  - `section`, `hgroup`, `details`, `summary`
  - `figure`, `figcaption`
- Altri elementi
  - `video`, `audio`, `canvas`, `embed`
  - `mark`, `command`, `output`, `time`
  - `progress`, `meter`, `datalist`

http://dev.w3.org/html5/html4-differences/

Multimedia

- `video, audio`
  - Supporto predefinito nelle specifiche (~ `img`), senza uso di plug-in proprietari
  - Dibattito su formati e codec (WebM vs. H.264)
- `canvas`: superficie libera, per disegno da script
- `embed`: altre risorse da includere, con plug-in
- `figure`: racchiude una `img` (o altro elemento) e una `figcaption` (didascalia)

Articoli e blocchi

- `article`
  - Un elemento principale, “auto-contenuto”
  - Un `article` può contenere altri `article`, es. commenti (annidati?) in un blog
- `nav`
  - Navigazione nel sito e/o nella pagina
- `header, footer`
  - Aree in alto e in basso nella pagina
- `aside`
  - Contenuto secondario, spesso x barre laterali

Sezioni

- `section` rappresenta una sezione generica
  - Dentro un `article` (come capitoli)...
  - O fuori da `article` (sez. di pagina)
  - Possono essere **annidate** (x struttura doc.)
- `hgroup`, area di titolo per ciascuna sezione
  - Dentro: `h1`, `h2`, per titolo e sottotitolo
- `details`, per introduzione alla sezione, o per altro contenuto espandibile
  - Normalmente visualizzato solo `summary`

Form HTML

- Sezione di documento tra tag `form`, contiene
  - Testo normale e markup
  - Elementi speciali chiamati controlli
    (aree per testo, opzioni, checkbox ecc.)
  - Etichette per questi controlli
- Gli utenti di solito “completano” questi controlli
  - Inserimento testo, selezione opzioni ecc.
- Infine il form viene sottomesso per l'elaborazione remota su un web server

Azione e metodo

- Il tag `form` prevede due attributi
- `action`: url a cui inviare i dati del form
- `method`: metodo HTTP da usare nell'invio dei dati del form (`get` o `post`, accordo con server)

```
<form
      action="http://xyz.com/script.cgi"
      method="post">
   …
</form>
```

Coppie nome/valore

- Ogni controllo di input in un form ha:
  - Un nome, definito dall'attributo `name` dei tag `input`, `select`, o `textarea`
  - Un valore, che l'utente imposta immettendo testo o agendo col mouse
- Dati inviati come insieme di coppie nome/valore
  - Campi di testo vuoti: inviata coppia nome/valore, valore = stringa vuota
  - Checkbox e pulsanti radio non selezionati non vengono proprio inviati

Tag per i controlli

- `input`
  - Controlli per l'immissione di testo, controlli "checkbox" e "radio", bottoni normali
- `select`
  - Selezione tra diverse opzioni, a discesa
- `textarea`
  - Campi per l'immissione di testo su più righe

Controlli input

- `input` crea diversi tipi di controlli, a seconda del valore dell'attributo `type`
  - `text`: campo per l'immissione di testo (default)
  - `password`: come text, ma nasconde il testo
  - `checkbox`: casella con segno di spunta, on/off
  - `radio`: per scegliere tra diverse opzioni
  - `submit`: bottone per inviare i dati inseriti
  - `reset`: buttone che ripristina la condiz. iniziale
  - `image`: immagine come bottone di submit
  - `hidden`: ulteriori coppie nome/valore da inviare al server, non visualizzate a utente

Attributi del tag input

- Insieme di attributi variabile a seconda di `type`
- `type` = `text` / `password`
  - `value`: valore iniziale per il campo
  - `size`: dimensione visualizzata del campo
  - `maxlength`: quantità di dati inseribili
- `type` = `checkbox` / `radio`
  - `value`: valore che ha il campo quando è selezionato; default = "`on`"
  - `checked`: se il campo per default è selezionato
- `type` = `submit` / `reset`
  - `value`: etichetta del bottone

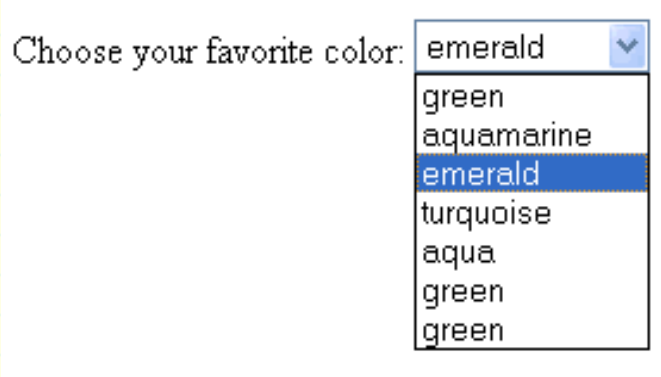
Tag input

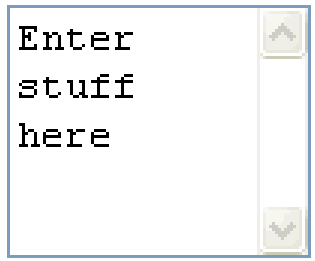
```
<form action="http://myhost.com/myscript.cgi" method="post">
State: <input type="text" name="state" value="IT" size="2"
maxlength="2"/>
<br/>Password: <input type="password" name="password"/>
<br/><input type="checkbox" name="moreinfo" value="yes"
checked="checked"/>Send me more info.
<br/>Select your gender below:
<br/><input type="radio" name="gender" value="F"/>Female
<br/><input type="radio" name="gender" value="M"/>Male
<br/><input type="submit" name="ok" value="OK, submit!"/>
<br/><input type="reset" value="Whoops - erase that"/>
<input type="image" src="hand.gif"/>
<input type="hidden" name="totalsofar" value="1290.65"/>
</form>
```

State: CA
Password:
Send me more info.
Select your gender below:
Female
Male
OK, submit!
Whoops - erase that

Tag select e textarea

```
Choose your favorite color:
<select name="favcolor">
    <option>green</option>
    <option>aquamarine</option>
    <option selected="selected">
      emerald</option>
    <option>turquoise</option>
    <option>aqua</option>
    <option value="green2">green</option>
    <option value="green3">green</option>
</select>

<textarea name="stuff" rows="5" cols="10">
  Enter stuff here
</textarea>
```





Input (e output) in Html5

- Nuovi *tipi* di input: `email`, `url`, `number`, `range`, `date`, `datetime`, `search`, `color`
- Attributo `placeholder`
  - Testo che scompare all'acquisizione del focus
- Attributo `list`, con *id* di un elemento `datalist`
  - Lista di suggerimenti (elementi `option`)
- `progress`, barra di progresso di una attività
- `meter`, indicatore di livello, tipo carburante

HyperText Transfer Protocol

- Protocollo di livello applicazione per sistemi informativi distribuiti, collaborativi e ipertestuali
  - Usato nel World-Wide Web fin dal 1990
  - Semplice & flessibile, vari tipi di dati
  - Senza connessione: una richiesta/risposta per ogni connessione
  - Senza stato: lo stato non è conservato tra connessioni diverse
  - Usa TCP/IP, porta 80; ma richiede solo un livello di trasporto affidabile

Versioni di HTTP

- Http/0.9 – Prima versione del protocollo
  - Semplice, trasferim. dati grezzi su Internet
- Http/1.0 – Miglioramenti, definito in RFC 1945
  - Formato messaggi simile a Mime (e-mail)
  - Meta-informazioni sui dati trasferiti
  - Modificatori per significato dei messaggi
- Http/1.1 – Connessione persistente, RFC 2616
  - Stessa connessione per più scambi richiesta/risposta
  - Possibile chiuderla per varie ragioni

Client e server

Un client invia una richiesta al server:

- Riga con metodo, url, versione del protocollo
- Intestazioni tipo Mime:

  Modificatori di richiesta

  Informazioni sul client
- Possibile corpo della richiesta (es. dati form)

Il server invia in risposta:

- Riga di stato con versione del protocollo; codice successo o errore
- Intestazioni tipo Mime:

  Informazioni sul server

  Meta-informazioni sulla entità (doc o altra risorsa)
- Possibile corpo per contenuto dell'entità richiesta

Generico messaggio

- Una riga di apertura (start-line)
- Zero o più campi di intestazione (header)
  - In formato RFC 822 (email)
  - general-, request-, response-, entity-header
- Una riga vuota che indica la fine dell'header
- Un possibile corpo del messaggio (body)
  - Per trasportare la risorsa (entità) associata alla richiesta o risposta

Metodo GET

Recupera l'informazione (in forma di entità) identificata dall'url della richiesta

- Se url = processo di produzione dati, questi dati sono restituiti come corpo della risposta
- Semantica modificata in GET condizionale se tra gli header:
  - If-Modified-Since, If-Unmodified-Since, ...
  - Per evitare spreco di risorse di rete

Metodo POST

- Copre diverse funzioni generali
  - Annotazione di risorse esistenti
  - Invio di messaggi a bulletin board, mailing list, newsgroup, o gruppi di articoli simili
  - Aggiunta di dati ad un database
  - Invio dati (es. form) a processo di gestione
- Url = processo che gestirà l'entità acclusa
  - Applicaz. server, o gateway per altri protocolli
  - Entità separata che accetta annotazioni

Connessioni usa-e-getta

- Il browser apre una connessione col server
  - Trasmette la parola “GET”, seguita da spazio, percorso per la risorsa, poi a-capo. Es.:

    ```
    GET /
    ```
- Il server risponde:
  - Riga di stato, intestazioni varie, una riga vuota e il contenuto del file richiesto
- Quindi chiude la connessione (one-shot)
- Il client deve poi aprire altre connessioni...
  - Per ricevere immagini, video...
  - Una per ogni oggetto necessario

Esempio di richiesta

```
POST /beta.jsp HTTP/1.1
Referer: http://www.alpha.com/alpha.jsp
Connection: Keep-Alive
User-Agent: Mozilla/4.61
Host: www.alpha.com:80
Cookie: name=value
Accept: image/gif, image/jpeg, */*
Accept-Language: en
[blank line here]
selected-item=1234&action=show+details
```

```
GET /beta.jsp?selected-item=1234&action=show+details HTTP/1.1
Referer: http://www.alpha.com/alpha.jsp
[...]
If-modified-since: Mon, 10 Jul 2000 22:55:23 GMT
[blank line here]
```

Esempio di risposta

```
HTTP/1.1 200 OK
Date: Fri, 12 Nov 2001 11:46:53 GMT
Server: Apache/1.3.3 (Unix)
Last-Modified: Mon, 12 Jul 2000 22:55:23 GMT
Accept-Ranges: bytes
Content-Length: 234
Content-Type: text/html
[blank line here]
<html>
    <head><title>Beta page</title></head>
    <body>
        <h1>Beta page</h1>...
```

Codici di stato

- 1xx – Informational
  - Richiesta ricevuta, elaborazione in corso
- 2xx – Success
  - Richiesta ricevuta, interpretata e accettata
- 3xx – Redirection
  - Necessario intraprendere azioni ulteriori
  - 301: Perm.; 302: Temp.; 304: Non modificata
- 4xx – Client Error
  - 400: Richiesta scorretta; 401: Non autorizzato; 404: Non trovato
- 5xx – Server Error
  - Server non può soddisfare richiesta valida

Cookie

I cookie sono un meccanismo generale per memorizzare e recuperare informazioni sul lato client di una connessione http

- Quando un server invia una risposta...
  - Può inviare anche informazione di stato
  - Da memorizzare sul client
  - Specificato il range di url per cui è valido
  - Può essere specificata una scadenza
- Poi, per ogni richiesta dal client in quel range...
  - Verrà ritrasmessa anche info di stato

Meccanismo dei cookie

- Si può segnalare un cookie al client includendo un header `Set-Cookie` nella risposta HTTP

```
Set-Cookie: NAME=VALUE; expires=DATE;
path=PATH; domain=DOMAIN_NAME; secure
```

- Quando richiede una url, il browser la confronta con tutti i cookie (con il rispettivo range di url)
- Per ciascun cookie che corrisponde, alla richiesta HTTP viene aggiunta una coppia chiave/valore

```
Cookie: NAME1=VALUE1; NAME2=VALUE2 ...
```

Uso dei cookie

- L’aggiunta di informazione di stato, semplice e persistente, sul lato client estende in modo significativo le capacità di applicazioni web
- Semplice carrello per commercio elettronico:
  - Elenco prodotti selezionati memorizzato in cookie
- Spesso al client è inviato solo un session-id
  - Usato dal server come chiave in un repository locale delle sessioni attive
  - Per trovare l’informazione di stato per il client

Introduzione a PHP

- ~~Personal Home Page~~
  Php: Hypertext Preprocessor
- Linguaggio scripting lato server, come asp e jsp
- File PHP eseguiti sul server e poi restituiti al browser come semplice HTML
- Supporta molti DB, spesso usato con MySQL
- Open source

http://php.net http://php.net/manual/it/

http://www.apachefriends.org/it/xampp.html

Hello, world!

```
<html>
<body>
<?php
echo "Hello world";
?>
</body>
</html>
```

- File PHP:

  normali tag html

  + codice di script
- Gli script cominciano con `<?php`
- …e finiscono con `?>`

Sintassi

- PHP ha una sintassi simile al linguaggio C
  - Operatori aritmetici, logici, assegn., confronto
  - Istruzioni condizionali (`if`, `switch`)
  - E di ciclo (`while`, `do-while`, `for`)
- Variabili e stringhe
  - Le variabili cominciano col simbolo `$`
  - Non sono tipate, non devono essere dichiarate
  - Le stringhe si concatenano con l'operatore `.`

Variabili

```
<?php
$txt1 = "Hello, ";
$txt2 = "world!";
$var = $txt1 . $txt2;
$number = 16;
$var = $number;
?>
```

Array associativi

```php
<?php

// coppie chiave/valore; valori di qualsiasi tipo
// 1. chiavi numeriche

$names = array("Pietro", "Giovanni", "Dario");
$names[1] = "Gianni"; // Pietro, Gianni, Dario
$dario = $names[2];

// 2. chiavi di tipo stringa

$ages = array("Zac" => 32, "Tom" => 28, "Rob" => 31);
$ages["Gino"] = 24;
$rob_age = $ages["Rob"];
?>
```

Parametri della richiesta

```
<html>…
<form action="welcome.php" method="get">
Name: <input type="text" name="name" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
…</html>
```

```
<html>…
<p>Benvenuto <?php echo $_GET["name"]; ?>.</p>
<p>Hai <?php echo $_GET["age"]; ?> anni.</p>
…</html>
```

Parametri della richiesta

- Possibile passare parametri da url (GET)
- http://127.0.0.1/welcome.php?name=jimbo&age=27
- Se il metodo del form è post, allora si deve usare la varibile `$_POST`
- La variabile `$_REQUEST` include gli elementi di `$_GET`, `$_POST`, `$_COOKIE`

Connessione a database

```
<?php
$con = mysql_connect("localhost", "peter", "abc123");
if (! $con) {
    die('Could not connect: ' . mysql_error());
}
mysql_select_db("my_db", $con);

// some code

mysql_close($con);
?>
```

Aggiunta dati in database

```
<?php
$con = mysql_connect("localhost", "peter", "abc123");
if (! $con) {
    die('Could not connect: '.mysql_error());
}
mysql_select_db("my_db", $con);

$query = sprintf("INSERT INTO persone (name, age)
VALUES ('%s', %d)",
    mysql_real_escape_string($_POST["name"]),
    $_POST["age"]);

mysql_query($query);

mysql_close($con);
?>
```

Lettura da database

```
<?php
$con = mysql_connect("localhost", "peter", "abc123");
mysql_select_db("my_db", $con);
$result = mysql_query("SELECT * FROM users");
$data = "";
while($row = mysql_fetch_array($result)) {
  $data .= "<tr><td>" . $row['name'] . "</td> .
            "<td>" . $row['age'] . "</td></tr>";
}
mysql_close($con);
?>

<table>
  <tr><th>Nome</th><th>Eta'</th></tr>
  <?php echo $data ?>
</table>
```