

Lecture 20

*408/508 Computational  
Techniques for Linguists*

# Last Time

I added ?tomato in manually on the URL line  
We will define a <form> this time!

- **Note:** slides corrected for Lecture 19 wrt. macOS configuration files
- Apple Perl executable (brew.sh differences):
  - M1/M2 Mac: #!/opt/homebrew/bin/perl
  - Intel Mac: #!/opt/local/bin/perl

# Today's Topic

- Sending form information to a Webserver program.
- Two methods:
  1. GET method
  2. POST method

# A worked example: CMUDict

- Let's define a <form> with a **GET** action

File: cmudictform.html

```
1<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">|
2<html>|
3<head>|
4<title>Using CMUDict from Perl</title>|
5<style>div {font-size: x-large}</style>|
6</head>|
7<body>|
8<h1>Using CMUDict from Perl</h1>|
9<div>|
10<form action="http://localhost/~sandiway/cmudict2.cgi" method="GET">|
11 Word: <input type="text" size="20" name="word">|
12 <input type="submit" value="Look up it!">|
13</form>|
14</div>|
15'Arpabet: <a href="https://en.wikipedia.org/wiki/ARPABET#Symbols">symbol table</a>|
16</body>|
17</html>|
```



# A worked example: CMUDict

File: cmudictform.html

```
1 <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">¶
2 <html>¶
3 <head>¶
4 <title>Using CMUDict from Perl</title>¶
5 <style>div {font-size: x-large}</style>¶
6 </head>¶
7 <body>¶
8 <h1>Using CMUDict from Perl</h1>¶
9 <div>¶
10 <form action="http://localhost/~sandiway/cmudict2.cgi" method="GET">
11 Word: <input type="text" size="20" name="word">¶
12 <input type="submit" value="Look up it!">¶
13 </form>¶
14 </div>¶
15 'Arpabet: <a href="https://en.wikipedia.org/wiki/ARPABET#Symbols">symbol table</a>¶
16 </body>¶
17 </html>¶
```

- the HTML form can be placed anywhere,
  - e.g.
  - ~/Sites (macOS)
  - ~/public\_html (Ubuntu)
- or
- /Library/WebServer/Documents/ (macOS)
  - /var/www/html/ (Ubuntu)

# A worked example: CMUDict

- File: cmudict2.perl
- Permissions: chmod a+x cmudict2.cgi

```
1#!/opt/homebrew/bin/perl ← customize /usr/bin/perl or /opt/local/bin/perl etc.  
2use Lingua::EN::CMUDict;↑  
3my $obj = new Lingua::EN::CMUDict;↑  
4my $string = $ENV{QUERY_STRING};↑  
5my $word = $string ? substr $string, 5 : $ARGV[0]; # remove word=↑  
6my $n = $obj->number_of_syllables($word);↑  
7print "Content-type: text/html; charset=utf-8\n\n";↑  
8print '<html><head><style>div {font-size: x-large}</style></head>';↑  
9print '<body><h1>Using CMUDict from Perl</h1><div>';↑  
10if ($n) {↑  
11  print "<em>$word</em> has $n syllable(s).</div>";↑  
12  my $pron = $obj->get_word($word);↑  
13  print "$pron<br>";↑  
14} else {↑  
15  print "<em>$word</em> not in cmudict</div>";↑  
16}↑  
17print 'Arpabet: <a href="https://en.wikipedia.org/wiki/ARPABET#Symbols">symbol table</a>';↑  
18print "</body></html>\n";↑
```

With a <form>  
with input field *word*,  
\$QUERY\_STRING contains  
*word=name*

# Perl substr function

[functions / substr \(source, CPAN\)](#)

**substr EXPR,OFFSET,LENGTH,REPLACEMENT**

**substr EXPR,OFFSET,LENGTH**

**substr EXPR,OFFSET**

Extracts a substring out of EXPR and returns it. First character is at offset zero. If OFFSET is negative, starts that far back from the end of the string. If LENGTH is omitted, returns everything through the end of the string. If LENGTH is negative, leaves that many characters off the end of the string.

```
my $s = "The black cat climbed the green tree";
my $color = substr $s, 4, 5;      # black
my $middle = substr $s, 4, -11;    # black cat climbed the
my $end    = substr $s, 14;        # climbed the green tree
my $tail   = substr $s, -4;       # tree
my $z      = substr $s, -4, 2;     # tr
```

# A worked example: CMUDict

The screenshot shows a web browser window with the URL `file:///Users/sandiway/courses/cmudict2.cgi`. The page title is **Using CMUDict from Perl**. There is a form with a text input field containing `tomato` and a submit button labeled `Look up it!`. Below the form, there is a link labeled `'Arpabet: symbol table'`.

```
<form  
action="http://localhost/~sandiway/cmudict2.cgi"  
method="GET">  
    Word: <input type="text" size="20" name="word">  
    <input type="submit" value="Look up it!">  
</form>
```

- Apache2 webserver receives:
  - `http://localhost/~sandiway/cmudict2.cgi?word=tomato`
- Webserver sets `$QUERY_STRING`:
  - `word=tomato`
- Webserver runs the program:
  - `~/Sites/cmudict2.cgi` (macOS)
  - `~/public_html/cmudict2.cgi` (Ubuntu)

# A worked example: CMUDict



- This is the Perl-computed response
- by the Webserver to the client Browser

# A worked example: CMUDict

- **Note:** the file
  - cmudict2.cgi could also be placed in /Library/WebServer/CGI-Executables
- **Then**
  - <form action="<http://localhost/cgi-bin/cmudict2.cgi>" method="GET">
- Webserver runs the program:
  - /Library/WebServer/CGI-Executables/cmudict2.cgi (macOS)
  - /var/www/html/cmudict2.cgi (Ubuntu)

# Sending information using GET

First:  Last:

- HTML form:

1. <form action="http://localhost/cgi-bin/get.cgi" method="GET">
2. First: <input type="text" name="first" size=12>
3. Last: <input type="text" name="last" size=12>
4. <input type="submit">
5. </form>

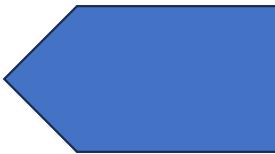
`http://localhost/cgi-bin/get.cgi?first=Sandiway&last=Fong`

- Information encoded using alphanumeric characters: why?
- URLs are restricted to alphanumeric characters only
- **bash** accesses the URL-encoded string via the environment variable **QUERY\_STRING**

Character	URL Encoded
;	%3B
?	%3F
/	%2F
:	%3A
#	%23
&	%26
=	%3D
+	%2B
\$	%24
,	%2C
<space>	%20 or +
%	%25
<	%3C
>	%3E
-	%7E
%	%25

# Today's Topic

- Sending form information to a Webserver program.
- Two methods:
  1. **GET method**
  2. **POST method**



# Sending information using POST

First:  Last:

- File: `form-post.html` could be in `/Library/WebServer/Documents`
- URL: `http://localhost/form-post.html`
- HTML form:
  1. `<form action="http://localhost/cgi-bin/read.cgi" method="POST">`
  2. First: `<input type="text" name="first" size=12>`
  3. Last: `<input type="text" name="last" size=12>`
  4. `<input type="submit">`
  5. `</form>`
- **bash** accesses the URL-encoded string via command `read`
  - cf. GET using `QUERY_STRING`

# Sending information using POST

- **bash** accesses the URL-encoded string on standard input via `read`

- `read.cgi` in `/Library/WebServer/CGI-Executables`:

```
1.#!/bin/bash
2.echo "Content-Type: text/plain"
3.echo
4.read input
5.origIFS=$IFS
6.IFS='=&'
7.set -- $input
8.IFS=$origIFS
9.echo "Form data \$2:<$2> \$4:<$4>"
```

## read

Read a line from standard input

### Syntax

```
read [-ers] [-a aname] [-p prompt] [-t timeout]
      [-n nchars] [-d delim] [name...]
```

IFS = **internal field separator** (for arguments)

default: space newline tab

set it to = and & because

**first=String1&last=String2**

## set -- *String*

-- option: positional parameters **\$1, \$2,..etc.** are set after splitting *String*

# Sending information using POST

- Client-side code:

```
1<!DOCTYPE HTML>|
2<html>|
3  <head>|
4    <title>CGI POST Example</title>|
5  </head>|
6  <body>|
7    <h1>CGI POST Example</h1>|
8    <form action="http://localhost/cgi-bin/read.cgi" method="POST">
9      First: <input type="text" name="first" size=12>|
10     Last: <input type="text" name="last" size=12>|
11     <input type="submit">|
12   </form>|
13 </body>|
14</html>|
```

form-post.html

place in

macOS: /Library/Webserver/Documents/  
Ubuntu: /var/www/html

- Server-side code:

```
1.#!/bin/bash
2.echo "Content-Type: text/plain"
3.echo
4.read input
5.origIFS=$IFS
6.IFS='=&'
7.set -- $input
8.IFS=$origIFS
9.echo "Form data \$2:<$2> \$4:<$4>"
```

read.cgi

place in

/Library/Webserver/CGI-Executables/  
Ubuntu: /usr/lib/cgi-bin

Sending  
information  
using POST

CGI POST Example

First:  Last:

localhost/cgi-bin/read.cgi

Form data \$2:<George> \$4:<Washington>

first=George&last=Washington  
\$1 \$2 \$3 \$4

# Example: adding names to a list on a server

- *Normally, you'd set up relational database software, e.g. mysql, on the webserver*

A screenshot of a Mac OS X desktop browser window. The address bar shows "file:///Users/sandiway/course". The main content area has a title "CGI GET Example" and a paragraph: "Adds names to a list kept on the webserver. Prints out the saved list." Below this is a form with fields: "First: ", "Last: ", and a "Submit" button.

A screenshot of a browser window with the address bar showing "localhost/~sandiway/get2.cgi". The main content area has a title "Table of Stored Names" and a table with two columns: "First" and "Last". The table contains the following data:

First	Last
A	BBB
B	CCC
X	YYY
asdasd	asdasd
tim	adas
tom	adas
david	adas
Michael	Anon
Michelle	Anon

# Example: adding names to a list on a server

**Client side:**  
addnames.html

**Server side:**  
get2.cgi  
names.txt

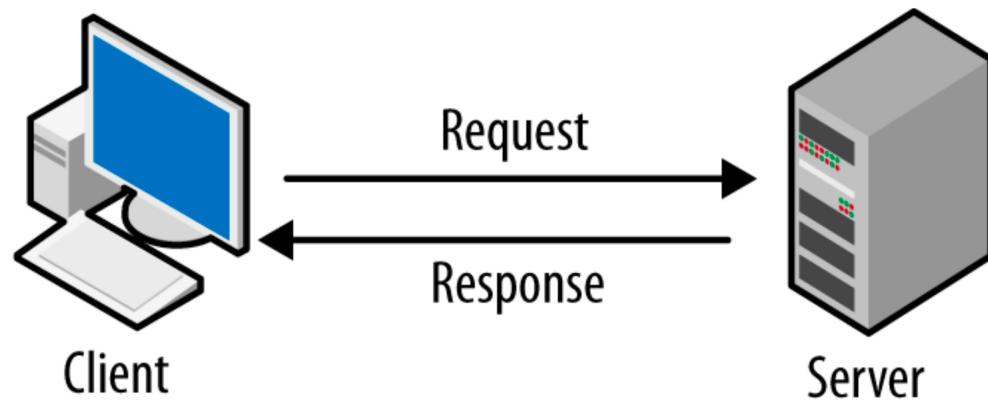


image from [https://madooei.github.io/cs421\\_sp20\\_homepage/client-server-app/](https://madooei.github.io/cs421_sp20_homepage/client-server-app/)

# A Note on Permissions

- **Caution!**
  - the Apache2.4 processes must be able to write to the directory in which `names.txt` is stored
  - this is not you, but typically the user is `_www` or `www`.
- On macOS:
  - `~/Sites` must have world write permission,
  - `chmod a+w ~/Sites`
- On Ubuntu:
  - `~/public_html` must have world write permission,
  - `chmod a+w ~/public_html`

# Example: adding names to a list on a server

## Server-side: get2.cgi

```
1#!/bin/bash
2echo "Content-Type: text/html; charset=utf-8"
3echo<
4echo "<html><head><style>"|
5echo "table { border-collapse: collapse }"|
6echo "td, th { border: 1px solid; padding: 4px }"|
7echo "th { background-color: lightyellow }"|
8echo "tr:nth-child(even) { background-color: #ccc }"|
9echo "</style></head>"|
10echo "<body><h1>Table of Stored Names</h1>"|
11echo "<form action=\"http://localhost/~sandiway/get2.cgi\" method=\"GET\">"|
12echo "First: <input type=\"text\" name=\"first\" size=12>"|
13echo "Last: <input type=\"text\" name=\"last\" size=12>"|
14echo "<input type=\"submit\"></form>"|
15origIFS=$IFS|
16IFS='=&'|
17set -- $QUERY_STRING|
18IFS=$origIFS|
19# names.txt must exist and
20# be write-able for everyone (www)|
21echo "$2 $4" >> names.txt|
22echo "<table><tr><th>First</th><th>Last</th></tr>"|
23while read -r first last|
24do|
25  echo "<tr><td>$first</td><td>$last</td></tr>"|
26done < names.txt|
27echo "</table>"|
28echo "</body></html>"|
29exit 0
```

## Client-side: addnames.html

```
1 <!DOCTYPE HTML>|
2 <html>|
3   <head>|
4     <title>CGI GET Example</title>|
5   </head>|
6   <body>|
7     <h1>CGI GET Example</h1>|
8     Adds names to a list kept on the webserver.|
9     <br>|
10    Prints out the saved list.|
11    <p></p>|
12    <form action="http://localhost/~sandiway/get2.cgi" method="GET">|
13      First: <input type="text" name="first" size=12>|
14      Last: <input type="text" name="last" size=12>|
15      <input type="submit">|
16    </form>|
17  </body>|
18 </html>|
```

**Create database file:** in ~Sites

touch names.txt      (*create an empty file*)  
chmod a+w names.txt

# Example: adding names to a list on a server

- Terminal (in ~/Sites):

```
Sites$ touch names.txt  
Sites$ chmod a+w names.txt  
Sites$ ls -l names.txt  
-rw-rw-rw- 1 sandiway staff 0  
Nov  8 19:46 names.txt
```

- After Submit is pressed:

```
Sites$ ls -l names.txt  
-rw-rw-rw- 1 sandiway staff 18  
Nov  8 19:49 names.txt  
(base) Sites$ cat names.txt  
George Washington
```

The image contains two screenshots of web browsers. The top screenshot shows a browser window with the title "file:///Users/sandiway/cours" and the heading "CGI GET Example". It displays the text: "Adds names to a list kept on the webserver. Prints out the saved list." Below this are input fields for "First" (containing "George") and "Last" (containing "Washington"), followed by a "Submit" button. The bottom screenshot shows a browser window with the title "localhost/~sandiway/get2.c" and the heading "Table of Stored Names". It displays a table with two rows: "First" and "Last" (both highlighted in yellow), and "George" and "Washington" respectively.

First	Last
George	Washington

# Example: adding names to a list on a server

A screenshot of a web browser window titled "localhost/~sandiway/get2.c". The page displays a heading "Table of Stored Names". Below it is a form with fields for "First" (containing "John") and "Last" (containing "Adams"). A "Submit" button is visible. To the right of the form is a table:

First	Last
George	Washington

A screenshot of a web browser window titled "localhost/~sandiway/get2.c". The page displays a heading "Table of Stored Names". Below it is a form with empty "First" and "Last" fields. To the right of the form is a table:

First	Last
George	Washington
John	Adams

```
Sites$ ls -l names.txt
-rw-rw-rw- 1 sandiway staff 29 Nov  8 19:53 names.txt
Sites$ cat names.txt
George Washington
John Adams
```

# Example: adding names to a list on a server

The screenshot shows a web browser window titled "localhost/~sandiway/get2.cgi". The main content area displays a table titled "Table of Stored Names" with two input fields ("First:" and "Last:") and a "Submit" button. The table contains two rows of data: George Washington and John Adams. The Web Inspector panel on the right shows the HTML and CSS code for the page. A blue arrow points from a callout box to the CSS rule for even-numbered rows.

**Table of Stored Names**

First:  Last:  Submit

First	Last
George	Washington
John	Adams

Web Inspector — localhost — get2.cgi

Elements    Console    Sources    Network    Timeline

E html > E body

```
<html>
  <head>
    <style>
      table { border-collapse: collapse }
      td, th { border: 1px solid; padding: 4px }
      th { background-color: lightyellow }
      tr:nth-child(even) { background-color: #ccc }
    </style>
  </head>
<body> = $0
  <h1>Table of Stored Names</h1>
  > <form action="http://localhost/~sandiway/get2.cgi" method="GET">...</form>
  > <table>
    > <tbody>
      > <tr>
        > <th>First</th>
        > <th>Last</th>
      > </tr>
      > <tr>
        > <td>George</td>
        > <td>Washington</td>
      > </tr>
      > <tr>
        > <td>John</td>
        > <td>Adams</td>
      > </tr>
    > </tbody>
  > </table>
</body>
</html>
```

CSS: grey background  
for even numbered  
rows

# Example: adding/deleting names on a server

**Table of Stored Names**

First:  Last:  Add:  Submit

First	Last	
First	Last	<input type="button" value="Delete"/>
John	A	<input type="button" value="Delete"/>
Mary	B	<input type="button" value="Delete"/>

**Client side:**  
addnames3.html

**Server side:**  
get3.cgi  
names.txt

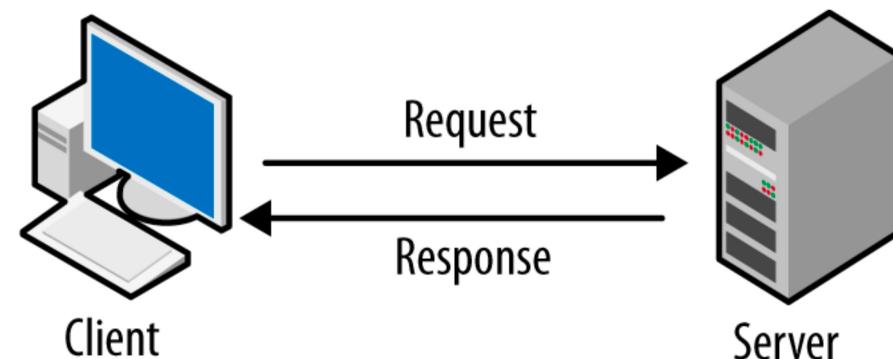


image from [https://madooei.github.io/cs421\\_sp20\\_homepage/client-server-app/](https://madooei.github.io/cs421_sp20_homepage/client-server-app/)

# addnames3.html

```
1<!DOCTYPE HTML>¶
2<html>¶
3  <head>¶
4    <title>CGI GET Example</title>¶
5  </head>¶
6  <body>¶
7    <h1>CGI GET Example</h1>¶
8    Adds names to a list kept on the webserver. Can also delete names.¶
9    <br>¶
10   Prints out the saved list.¶
11   <p></p>¶
12   <form action="http://localhost/~sandiway/get3.cgi" method="GET"> ¶
13     First: <input type="text" name="first" size=12>¶
14     Last: <input type="text" name="last" size=12>¶
15     Add: <input type="checkbox" name="add" checked>¶
16     <input type="submit">¶
17   </form>¶
18 </body>¶
19</html>¶
```

file:///Users/sandiway/courses/ling508-20/addnames3.html

## CGI GET Example

Adds names to a list kept on the webserver. Can also delete names.  
Prints out the saved list.

First:  Last:  Add:  Submit

\$1 \$2 \$3 \$4 \$5 \$6

**Note:** actually \$1, \$3 and \$5 are names "first", "last" and "add"

# get3.cgi

localhost/~sandeway/get3.cgi?first=111&last=111

Table of Stored Names

First:  Last:  Add:  Submit

First	Last
First	Last

if empty, will just list the current table

```
<html><head><style>
table { border-collapse: collapse }
td, th { border: 1px solid; padding: 4px }
th { background-color: lightyellow }
tr:nth-child(even) { background-color: #ccc }
</style>
<script>function del(r){
var tr=document.getElementsByTagName("table")[0].rows[r];
document.getElementById("f1").first.value = tr.cells[0].innerText;
document.getElementById("f1").last.value = tr.cells[1].innerText;
document.getElementById("f1").add.checked = false;
}</script></head><body><h1>Table of Stored Names</h1>
<form id="f1" action="http://localhost/~sandeway/get3.cgi" method="GET">
First: <input type="text" name=first size=12>
Last: <input type="text" name=last size=12>
Add: <input type="checkbox" name="add" checked>
<input type="submit"/></form>
<table><tr><th>First</th><th>Last</th></tr>
<tr><td>First</td><td>Last</td><td><button onclick="del(1)">Delete</button></td>
</tr>
</table>
</body></html>
```

Note: action is <http://localhost/~sandeway/get3.cgi>

# get3.cgi

The screenshot shows a web browser window with the URL `localhost/~sandaway/get3.cgi?first=&last=&add=on`. The title bar says "Table of Stored Names". The page contains a form with fields for First name and Last name, and a checked "Add" checkbox. Below the form is a table with two rows of data, each with a "Delete" button.

First	Last	
George	Washington	<input type="button" value="Delete"/>
John	Adams	<input type="button" value="Delete"/>

- Current state of names.txt

# get3.cgi

The screenshot shows a web browser window for `localhost/~sandiv/get3.cgi`. The page title is "Table of Stored Names". It contains a form with fields for "First:" and "Last:", an "Add:" checkbox (unchecked), and a "Submit" button. Below the form is a table with two rows:

First	Last	
George	Washington	<button>Delete</button>
John	Adams	<button>Delete</button>

The Web Inspector window shows the source code for the page. The `del(r)` function is defined in the head section:

```
function del(r){  
    var tr=document.getElementsByName("table")[0].rows[r];  
    document.getElementById("f").first.value = tr.cells[0].innerText;  
    document.getElementById("f").last.value = tr.cells[1].innerText;  
    document.getElementById("f").add.checked = false;  
}
```

Arrows from three callout boxes point to the delete buttons in the table:

- An arrow points to the first delete button with the text "del(row) function".
- An arrow points to the second delete button with the text "delete button calls del(1)".
- An arrow points to the third delete button with the text "delete button calls del(2)".

The browser's address bar shows the URL `localhost/~sandiv/get3.cgi`.

# get3.cgi

localhost/~sandiway/get3.cgi?first=George&last=Washington

**Table of Stored Names**

First: George Last: Washington Add:  Submit

First	Last	
George	Washington	<input type="button" value="Delete"/>
John	Adams	<input type="button" value="Delete"/>

press Submit

press Delete  
name is entered in the form, Add is checked

localhost/~sandiway/get3.cgi?first=George&last=Was

**Table of Stored Names**

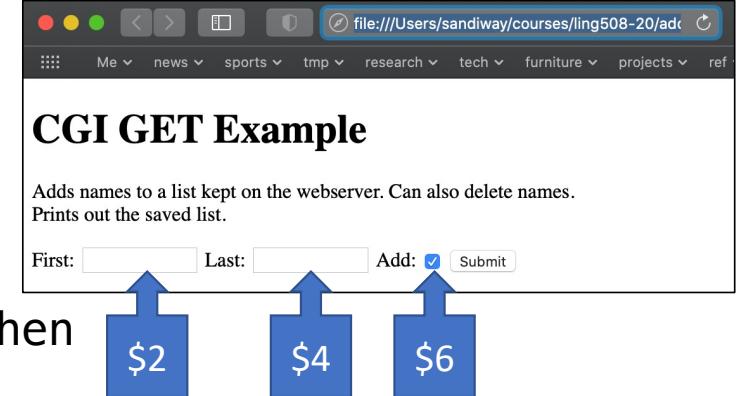
First:  Last:  Add:  Submit

First	Last	
John	Adams	<input type="button" value="Delete"/>

<http://localhost/~sandiway/get3.cgi?first=George&last=Washington>

# get3.cgi

```
if [ "$6" = 'on' ]; then
    if [ "$2" != '' ] && [ "$4" != '' ]; then
        echo "$2 $4" >> names.txt
    fi
else
    perl -i -ne "print unless m/^$2 $4$/" names.txt
fi
```

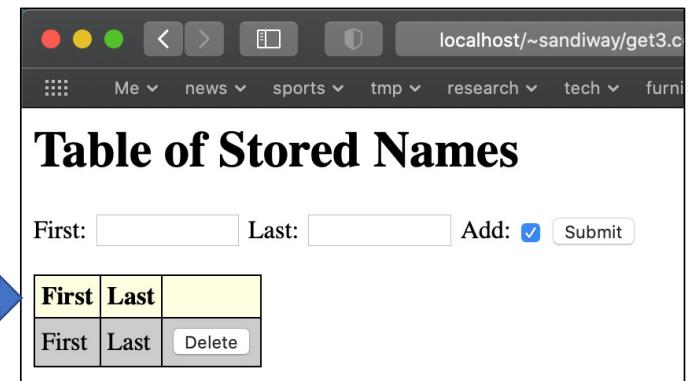
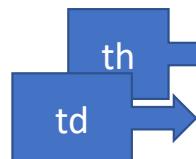


## Using the `-i` Option   perl `-i`

The `-i` option lets you modify files in-place. This means that Perl will automatically rename the input file and open the output file using the original name. You can force Perl to create a backup file by specifying a file extension for the backup file immediately after the `-i`. For example, `-i.bak`. If no extension is specified, no backup file will be kept.

# get3.cgi

```
echo "<table><tr><th>First</th><th>Last</th><th></th></tr>"  
i=1  
while read -r first last      -r      do not allow backslashes to escape any characters  
do  
    echo "<tr><td>$first</td><td>$last</td><td><button  
onclick=\"del($i)\">Delete</button></td></tr>"  
    ((i++))  
done < names.txt      < names.txt means read from file names.txt  
echo "</table>"  
echo "</body></html>"
```



# get3.cgi

```
□ | < > [get3.cgi] Response ▲  
1 <html><head><style>  
2 table { border-collapse: collapse }  
3 td, th { border: 1px solid; padding: 4px }  
4 th { background-color: lightyellow }  
5 tr:nth-child(even) { background-color: #ccc }  
6 </style>  
7 <script>function del(r){  
8 var tr=document.getElementsByTagName("table")[0].rows[r];  
9 document.getElementById("f").first.value = tr.cells[0].innerText;  
10 document.getElementById("f").last.value = tr.cells[1].innerText;  
11 document.getElementById("f").add.checked = false;  
12 }</script></head><body><h1>Table of Stored Names</h1>  
13 <form id="f" action="http://localhost/~sandiway/get3.cgi" method="GET">
```

localhost/~sandiway/get3.cgi

## Table of Stored Names

First:  Last:  Add:  Submit

First	Last	
First	Last	Delete

First:  Last:  Add:  Submit

first.value      last.value      checked = false