

CGI 5

Page Sequence

User choice request1 -> page1

User choice on page1: request2 -> page2

User choice on page2: request3 -> page3

Technique 1: hidden fields

Appear to give continuity to the pages using hidden fields to remember things from one request to the next

Back Button Problem -- Stale State

Problem: user can see old state

Imperfect solution: browser makes the request that led to the page again -- has its own problems

Back Button Problem -- Double Submit

Browser makes the request again

Request is like a command to the server, so we are sending the command again

What if the command was "add a record" or "delete a record" -- by reloading the page the command led to, we do the command again.

e.g. reload your record delete page, you should get the "that record already deleted" error

Note: it's ok if your hw3 suffers from this problem for all its operations -- we'll address this for HW4

How could you proof your CGI against double submit?

Typical Web Interface

Sequence of pages

Connected by requests -- each like a command

"Form Interfaces Are a Hack"

A sequence of HTML/Form pages, connected with URLs and form submits

Has lots of problems: double submits, can see old/stale state, and most important, the interaction is very coarse

We underestimate how poor this interface is since we're so used to it, but actually it's a huge step backwards compared to a typical GUI app

Or put another way, there's a lot of potential to do better

Also, it's a bit of a trap to take your GUI design ideas and just re-create them on the web.

Sample Problem 1: zip code

Consider the problem of letting the user put in their zip code

Idea -- Perl Variable

Let's just store it in a Perl variable in the CGI
No: the CGI runs fresh each time

Idea -- Write To File

Let's write it to a zip.txt file, and then we can read it next time
No: what about all the other users?

Idea -- By IP Address

We'll store it in zip.txt, with a separate line for each IP addr
No: can have multiple clients appearing to come from one IP addr

Idea -- Hidden Variables

This works, but with the problems above

Sample Problem 2: Login

1. Ask user for password
 2. Compare to server side password and it matches ...
- What do we return to the client?
Want them to be able to go through a sequence of pages now with the appearance of being logged in

Idea -- auth=yes

Could put an auth=yes hidden field in the reponse, and then check for it in later requests
No: client can just fake that -- edit the HTML or put it on the URL themselves

Idea -- auth=password

Could put auth=password in hidden field
Slightly better, but it's bad to leave the password around like that
"Library" problem: user 1 at library, shops some and then walks away. User 2 comes up to the computer and can see user 1's password