

Web Application Development

Lecture #10 – GET, POST, Sessions
February 13, 2008

Mark Stehlik
(with a nod to Jeff Eppinger)

Outline for Today

- GET & POST
- Hidden fields
- Cookies
- Sessions

HTTP requests

- Request/response
- GET
- POST
- (PUT, UPDATE)

HTTP GET

- GET
 - Asks the server to get a resource and send it back
 - If there are parameters, they appear in the URL, starting with a ? and separated by &
 - Limited in the amount of data that can be sent (and sent in the clear, right in the URL!)
 - Meant for getting things (with parameter to help determine what to get)
 - Idempotent - no side effects (well..., what about those link_to's)

HTTP GET Request Format

```
GET <identifier>?<query-string> HTTP/<version>  
<header-name>: <header-value>  
<header-name>: <header-value>  
...  
<header-name>: <header-value>  
<blank-line>
```

HTTP GET Request Example

GET /index.html HTTP/1.1

Accept: image/gif, image/x-bitmap, image/jpeg, image/pjpeg,
application/x-shockwave-flash, application/vnd.ms-excel,
application/vnd.ms-powerpoint, application/msword, */*

Accept-Language: es-us,en-us;q=0.5

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT
5.1; SV1; .NET CLR 1.1.4322)

Host: localhost

Connection: Keep-Alive

HTTP Get Request Example w/Param

GET /hello.html?name=Mark HTTP/1.1

Accept: image/gif, image/x-bitmap, image/jpeg, image/pjpeg,
application/x-shockwave-flash, application/vnd.ms-excel,
application/vnd.ms-powerpoint, application/msword, */*

Accept-Language: es-us,en-us;q=0.5

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT
5.1; SV1; .NET CLR 1.1.4322)

Host: localhost

Connection: Keep-Alive

HTTP POST

- POST
 - Used to send data to be processed
 - Think “updating” - changing something on the server
 - Parameters are in the body, the “payload”
 - Form submit
 - NOT idempotent (has side-effect – changes data)

HTTP Post Request Format

```
POST <identifier> HTTP/<version>
<header-name>: <header-value>
<header-name>: <header-value>
Content-Length: <message-length>
...
<header-name>: <header-value>
<blank-line>
<message-body>
```

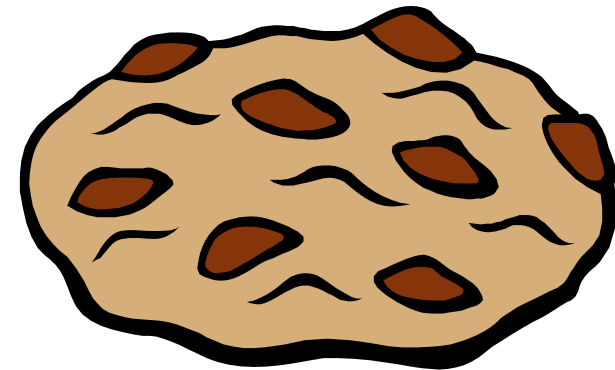
Hidden Fields in HTML

```
<input type="hidden" name="id" value="5"/>
```

- It's a type of field that you can put in an HTML form
- This field not displayed, but it is returned as a parameter on the subsequent request
- Is this secure?

Question for you?

What's a cookie?



Cookies in the web context

- A cookie is a piece of information that a web application hands to your browser
- The browser (and probably you) don't know how to interpret the cookie, but it hangs onto it
- The browser hands the app the cookie on each subsequent request
 - unless the cookie is deleted, expired, modified, etc
- Do you know who's been giving you cookies ?

Cookies don't always taste good

- Cookies can only contain small amounts of data
 - up to 4kb
- Cookies might not be accepted by the browser
 - do you accept cookies?
- Cookies optionally may be stored on disk
- Cookies can be modified by the user
- Cookies can get stale or thrown out
 - (they expire or are deleted)

Cookie Examples

- Sites with a “Remember me on this computer” check box
- Sites that track your activity and preferences:
 - amazon.com
 - cnn.com

Sessions

- A way to keep track of what the user has done during this visit to your web application
- Rails uses cookie data to match a request with session information (all key/value pairs from that user) stored on the server
- So if you're running a Rails app, you'll need to accept cookies...
- And where's the session data stored?