Denne guide er oprindeligt
udgivet på Eksperten.dk

COMPUTERWORLD

POST vs. GET i praksis

Denne artikel har til formål at gøre læseren i stand til at afgøre, hvornår man skal benytte POST henholdsvis GET som forespøgselsmetode i forbindelse med HTML-formularer, links og andre forespørgsler. Inledningsvist præsenteres HTTP/1.1 overordnet.

Skrevet den **11. Feb 2009** af **jensgram** I kategorien **Programmering / Generelt** | ★★★☆☆

Introduktion
Fra tid til anden bliver der spurgt til, hvornår man bør benytte POST eller GET i en HTML-formular. Ligeledes sker det til tider, at en bruger brokker sig over, at dennes browser beder vedkommende bekræfte, at data skal gensendes for at opdatere en side (via POST). Det er mit mål, at det følgende skal give læseren indblik i filosofien bag POST- og GET-metoderne for derved selv at kunne foretage det rigtige valg (og forstå hvorfor browser beder om bekræftelse).

Hypertext Transfer Protocol
En forespørgsel via HTTP består af en header-sektion og en "krop" (body). Dette kan for så vidt sammenlignes med HTML, hvor metadata forefindes inden for <head>-tagget, mens selve dokumentets indhold er omsluttet af <body>-tagget. For HTTP-protokollen er syntaksen dog noget mere stringent for head-delen:
<Header>: <Data>, hvor hver header står på en linie for sig (afsluttet med CRLF, \r\n).

Head og body adskilles af en blank linie - dvs. to på hinanden følgende linieskift. En simpel forespørgsel på en side kunne se ud som:

```
HEAD / HTTP/1.1
Host: www.jensgram.dk
```

Bemærk, at "/" efter HEAD i første linie betyder, at jeg forespørger roden af sitet. Serveren ved, at indekssiden hedder index.php, men det kunne jeg også selv have skrevet. Læg desuden mærke til, at der er en blank linie efter Host-linien - den afslutter head-sektionen (og dermed forespørgslen i dette tilfælde, da HEAD-forspørgsler ikke har nogen body.

Gennem et forespørgselsværktøj (http://mbn.dk/q/?method=HEAD&url=http%3A%2F%2Fwww.jensgram.dk) afsløres serverens svar:

```
HTTP/1.1 200 OK
Date: Wed, 10 Jan 2007 12:40:53 GMT
Server: Apache/1.3.33 (Unix) PHP/4.3.11
[headers udeladt til fordel for læsbarhed...]
Content-Type: text/html; charset=iso-8859-1
```

Bemærk, at der heller ikke her er noget indhold i body-sektionen (det er netop formålet med HEAD-metoden, jf. næste afsnit).

For det følgende, skulle du nu besidde tilstrækkeligt kendskab til HTTP-forspørgslernes natur. Har du appetit på mere så kig i afsnittet "Links til videre læsning" nedenfor.

HTTP & forespørgselsmetoder
I HTTP-protokollen findes 8 forespørgselsmetoder, hvoraf kun følgende 3 er relevante for denne artikels fokus (HEAD er medtaget for ovenstående eksempels skyld):
1) **GET**: Den almindeligste metode, der beder om indholdet for en given adresse (URI)
2) **HEAD**: Som GET, dog kun headers (se eksempel ovenfor).
3) **POST**: Forespørger en ressource - body-sektionen kan indeholde data (fra formularer etc.)

I en GET-forespørgsel kan man medsende data i en såkaldt "query string". Ønsker jeg eksempelvis at søge efter "eksperten" på Google, kan jeg indtaste adressen http://www.google.com/search?q=eksperten i min browser, der ved sender en forespørgsel lignende:

```
GET /search?q=eksperten HTTP/1.1
Host: www.google.com
```

På hosten www.google.com forespørger man ressourcen search med argumentet "q=eksperten". Dette argument kunne eksempelvis tilgås via $_GET['q'] i PHP, men det er uden for denne artikels fokusområde. Vigtigt er det, at den resulterende side kan findes igen gennem den benyttede URI (hér: URL), da alle data er en del heraf. Dokumentet er således identificébart (selv om de enkelte hits naturligvis kan skifte).

Skulle ovenstående forespørgsel i stedet foretages via POST, ville det se ud som følger:

```
POST /search HTTP/1.1
Host: www.google.com
Content-Length: 11

q=eksperten
```

Her ses, at data nu ikke længere er en del af URL'en (http://www.google.com/search), men i stedet udgør forespørgslens body-sektion. Det betyder, at man ikke kan linke til netop denne forespørgsel (søgetermen er ikke en del af URL'en). Desuden svarer Google noget i retning af "The server is unable to process your request.", men det er mindre relevant for nærværende.

Sikre metoder
Sikker interaktion er i HTTP/1.1-specifikationen defineret som interaktion, hvor klienten ikke er "ansvarlig" for sine handlinger. Således er det at følge et link eller at foretage en simpel søgning på eksempelvis Google et eksempel på sikker interaktion. En transaktion via Netbank er derimod ikke sikker, da man som bruger netop kan drages til ansvar for de handlinger man foretager; handlinger, der rent faktisk har en effekt.

Af ovennævnte metoder er HEAD og GET defineret som sikre. Det betyder *ikke*, at indholdet på en forespurgt URL ikke må ændre sig, men blot at forespørgslen **ikke bør ændre serverens tilstand** - forespørgslen har ingen sideeffekter. I denne sammehæng er det vigtigt at forstå, at en GET-forespørgsel - eksempelvis http://www.google.com/search?q=eksperten - kan afstedkomme forskelligt indhold (eftersom

Google får indekseret nye sider, der har tilknytning til "eksperten"). Det essentielle er, at søgningen ikke forårsager ændringer på serveren.

POST-forespørgsler er derimod ikke sikre, da det netop er filosofien, at en sådan må ændre serverens tilstand (tilmelde en bruger til en mailing-liste e.l.). Problemet er imidlertid, at det er op til udviklerne at håndhæve GET-forespørgslens immanente egenskab som sikker metode. **Det er altså op til udvikleren at sørge for, at GET-forespørgsler forbliver sikre.**

Som en sidebemærkning kan det nævnes, at metoderne GET, HEAD, PUT, DELETE, OPTIONS og TRACE er idempotente. Det betyder, at sideeffekterne for gentagne, identiske forespørgsler er identiske med sideeffekterne for én forespørgsel. PUT og DELETE er idempotente, da en ressource i sagens natur kun kan oprettes eller slettes én gang (der ses bort fra fejlmeddelelser ifm. idempotens). POST er *ikke* idempotent, hvilket forklarer, at **en browser *bør* bede brugeren bekræfte at POST-data skal sendes igen.**

Hvilken metode skal jeg så vælge?
Med udgangspunkt i, at:
a) Enhver ressource er identificérbar via en URI (hér: URL, da vi beskæftiger os med HTTP), og
b) GET benyttes som defineret - i.e. som en sikker metode
skulle det nu være muligt at fastslå, hvornår man skal vælge POST henholdsvis GET.

Du bør benytte GET-metoden, hvis den ønskede interaktion skal tænkes som en forespørgsel på en ressource. Netop herfor er GET den mest benyttede forespørgselsmetode, da almindelig surfing ved at følge hyperlinks netop er en kæde af simple forespørgsler på ressourcer.

Du bør benytte POST-metoden, hvis interaktionen er af en sådan natur, at den kan afstedkomme ændringer på serveren. Det kunne eksempelvis være tilmelding til nyhedsbreve, overførsel af penge via Netbank, bestilling på webshop etc. Vær dog opmærksom på, at der er visse tilfælde, hvor det er acceptabelt at lade GET-forespørgsler ændre tilstand på serveren. Et eksempel kan være en besøgstæller, der inkrementeres for enhver forespørgsel. Den omstændighed, at brugeren ikke er ansvarlig for denne handling, gør, at forespørgslen som sådan stadig er sikker.

Vil det sige, at GET bør være forbeholdt links og POST forbeholdt formularer?
Nej! Det var absolut ikke min hensigt at foranledige læseren til at tro det. Ved at tage udgangspunkt i HTTP's definition, var det en del af mit mål at lade læseren forstå, hvordan både GET- og POST-forespørgsler reelt "ser ud". Dermed skulle det også stå klart, at et hyperlink ikke kan ligge til grund for en POST-forespørgsel.

Det var også en del af min målsætning at vise, at en GET-forespørgsel (og dermed et hyperlink) godt kan benyttes til at ændre tilstand på en server. Det er udviklerens ansvar, at dette ikke er tilfældet.

Hvis en formular ikke forårsager tilstandsændring på serveren, bør du benytte GET. Tænk eksempelvis på søgeformularen på Googles forside (http://www.google.com/), der benytter sig af GET. Det har den positive effekt, at brugere kan kopiere URL'en for en specifik søgning (som ovenfor). Et andet eksempel kunne være en artikel, der består af flere sider, hvor man vil kunne benytte en formular til at tilgå de enkelte sider.

Afslutning
Det er mit håb, at læseren nu har tilegnet sig grundigere forståelse af, hvornår POST før foretrækkes til fordel for GET og *vice versa*. I ovenstående har jeg ikke været inde på praktiske og sikkerhedsmæssige overvejelser, men appelerer i nogen grad til læserens sunde fornuft. Vigtigst er det nok at være opmærksom på, at det er en skrøne, at POST-forespørgsler skulle være mere sikre eller sværere at eftergøre end GET-forespørgsler. I samme omgang bør det nævnes, at følsomme data aldrig bør være del

af en URI.

Med venlig hilsen
- Jens Gram, http://www.jensgram.dk/

Links til videre læsning
- Hypertext Transfer Protocol på Wikipedia: http://en.wikipedia.org/wiki/Http
- RFC 2612 om HTTP/1.1 fra W3C/MIT: http://rfc.sunsite.dk/rfc/rfc2616.html
- URIs, Addressability, and the use of HTTP GET and POST fra W3C: http://www.w3.org/2001/tag/doc/whenToUseGet.html
- HTTP query tool af Morten Blinksbjerg Nielsen: http://mbn.dk/q/

Change log:

```
2007/01/10: Første version publiceret
2007/01/11: Tyrk-fjel rettet som påpeget af phliplip
2009/03/23: Links rettet til E5
```

Kommentar af phliplip d. 11. Jan 2007 | 1

God artikel! Men i ovennævnte sammenhæng skal _POST nok rettes til _GET "Men Dette argument kunne eksempelvis tilgås via $_POST['q'] i PHP"

;)

Kommentar af greew d. 25. Jan 2007 | 2

Meget brugbar!! Takker! - Jesper

Kommentar af amite d. 19. Feb 2007 | 3

Glimrende artikkel! Omhandlede præcist det den skulle. Man fik rusket op i nogle tanker på et område som man normalt ellers kan have en tendens til at hoppe (for?) let henover. /thumbs_up

Kommentar af wickedd d. 11. Jan 2007 | 4

God læsning!

Kommentar af danieru d. 13. Feb 2007 | 5

Jaja da! ;)

Kommentar af windcape d. 11. Jan 2007 | 6

jow jow

Kommentar af b1sket d. 16. Feb 2007 | 7