Query String
https://en.wikipedia.org/wiki/Query_string

Query strengur er hluti af URL:

```
http://www.domain.com:80/path/to/resource?id=5&name=Gunnar
```

http://	protocol
www.domain.com	host (hýsingaraðili)
80	port
/path/to/resource	resource path
?id=1&name=Gunnar	**query**

Það sem kemur á eftir spurningamerki **?** eru gögn, breytur sem innihalda gildi. Breytan **id** geymir gildið **5** og breytan **name** geymir gildið **Gunnar**. Þessi gögn eru sett fram í key/value pörum. **&** sértáknið er notað til að aðgreina pörin.

Hægt er að ímynda sér query string sem Python dictionary, sem runu af key-value pörum, en ólíkt dictionary þá geta sömu key komið fram oftar en einu sinni í sama query string.

Dæmi um löglegan query string í URL: `http://example.net?name=John&name=James`

Fleiri query string dæmi:

- somepage.php?page=2
- somepage.php?page=3&category=5
- http://google.com/search?q=php

Linkur með query string og encode.

Við þurfum að hafa gætur á sértáknum sem eru ekki í ASCII stafamenginu þegar við búum til linka með URL. **&** táknið (non-encoded ampersand) þarf t.d. að skipta út fyrir & til að linkur sé löglegur.

Dæmi: `http://www.domain.com/path?id=9&category=5`

Til að linkur sé löglegur þá:

```
<a href="http://www.domain.com/path?id=9&amp;category=5">linkur</a>
```

Önnur sértákn sem þarf að huga að eru t.d; tómt bil sem þarf að filla með **%20**

Dæmi John Smith:

```
<a href="http://example.net?name=John%20Smith">linkur</a>
```

Önnur sértákn og ASCII jafngildi þeirra eru t.d:

<	%3C	>	%3E	#	%23	%	%25	{	%7B	}	%7D
|	%7C	\	%5C	^	%5E	~	%7E	[	%5B	]	%5D
`	%60	;	%3B	/	%2F	?	%3F	:	%3A	@	%40
=	%3D	&	%26	$	%24	+	%2B	"	%22	space	%20

Til eru föll í forritunarmálum sem geta gert þessar umbreytingar fyrir þig, t.d. urlencode() í php og urllib module í python.

Nánar um ASCII (American Standard Code for Information Interchange)

https://en.wikipedia.org/wiki/ASCII

HTTP Request og Query Parameter

Þegar við smellum á link með query streng þá erum við að framkvæma **GET request**. Ásamt því að senda gögn til miðlara.

Miðlari notar þessi gögn venjulega til að ákvarða hvað sé framkvæmt með forriti sem við forritum (routing) og hverju sé skilað til biðlara (e. client) **HTTP Respond**.

Punktar varðandi GET requests:

- GET requests can be cached
- GET requests remain in the browser history
- GET requests can be bookmarked
- GET requests should never be used when dealing with sensitive data
- GET requests have length restrictions
- GET requests should be used only to retrieve data

Form getur einnig innihaldið query string. Með því að smella á submit hnapp í formi þá verður til **POST request**. Þar sem gögn úr forminu er safnað saman og sent til miðlara (nánar um það síðar)

Punktar varðandi POST request:

- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked

- POST requests have no restrictions on data length

Meðhöndlun á Query string.

Miðlari getur meðhöndlað HTTP request annað hvort með því að

a) lesa skrár í skráarkerfinu byggt á URL slóðinni eingöngu.

b) með notkun forrits / forritunarmáls til að vinna úr query string ásamt URL slóð.

Í PHP verða til innbyggð fylki / listar (associative array) sem innihalda query parametra (GET array) og gögn úr formi (POST array) sem við höfum aðgang að getum unnið frekar úr með php.

Í Python og fleiri málum þá er notast við söfn við slíka vinnslu (t.d. urllib í PSL, Requests Module (external) eða með framework eins og Bottle, Flask, Django osfrv.).

Routing og URL rewrite

Í stað þess að nota URLs (Uniform Resource Locator) eins og *index.php?xxx=111&yyy=222* þá nota flest framewok URLs á eftirfarandi hátt:

- example.com/songs
- example.com/songs/editsong/17
- example.com/subpage

  osfrv.

Til að ná þessu fram og losna þannig við að bæta handvirkt query strengjum við linka þá er hægt að notast við URL rewrite; sem beinir á annað url, felur .py endingar osfrv, regluar expression og Routing

,,Routes are basically paths the user takes which are attached to code that will be triggered when a user reaches the specific route.“

,, URL is a destination (address) and the route is how you navigate (path) to get there. (road)“

Dæmi um Routing útfærslu

Í stað þess að notast við skrárkerfi þá notum við föll (e. function) til að ákveða hvað sé framkvæmt og birt á vefinn.

Dæmi: example.com/songs/editsong/17

Fyrsti hlutinn á eftir domain (example.com) í urlinu, **songs** velur viðeigandi skrá sem er oft kallað controller (geta verið margir slíkir í vef)

Seinn hlutinn **editsong** er fallið (e. function) sem er valið í songs skránni. Fallið editsong sér þá um að uppfæra gögn í þessu tilfelli (t.d. á gagnagrunni) með SQL aðferðum sem og ná í þann kóða sem þarft til að birta gögnin á vefsíðu (t.d. html, python, css) sem geta verið vistuð í sér skrám.

17 er id gildið í query string sem var sent til að vita á hvaða link (lag) var smellt.

Ef eingöngu er slegið inn example.com þá gætum við útfært routing þannig að index() aðferð fari sjálfkrafa af stað og forsíðu er skilað til biðlara (e. client)

Þú getur þannig gert í raun hvað sem þú ákveður í föllunum.

Routing og Python
https://wiki.python.org/moin/Routing

Routing útfært í Bottle
https://bottlepy.org/docs/dev/tutorial.html#request-routing