

INF105 - 24
DB and WWW | Sysadmin tasks

David Grellscheid

Web apps med databaser

SQL i dynamiske sider server-side

- Kan bruke SQL server-side til å fylle ut templates
- Flask-eksempel fra forelesning 20:

```
def build_list():
    all_jsons = Path("movies").glob("*.json")
    movie_list = []
    for path in all_jsons:
        id = path.stem
        info = get_movie_info(path)
        movie_list.append((info['year'], info['name'], id))
    return movie_list
```

```
def build_list():
    res = cur.execute("SELECT year, title, id FROM movies")
    movie_list = res.fetchall()
    return movie_list
```

AJAX - pattern
client side updates

- Vi kan også sende DB info til klienten og la JavaScript gjøre jobben
- På serveren: definer en URL som returnerer JSON data
  (ligger ofte under "/api/")
- I JS: fetch(api-URL) og gjør noe med datasett
- eksempel: movies

AJAX - pattern
client side updates

```
@app.route('/actor/<int:id>')
def single_actor(id):
    id = escape(id)
    info = get_actor_info(id)
    return render_template('actor.html', **info)
```

vanlig URL som returnerer HTML basert på en template

```
@app.route('/api/moviecount/<int:actorid>')
def movie_count(actorid):
    res = cur.execute("SELECT count(DISTINCT movie_id) FROM actors WHERE actor_id=?", (actorid,))
    result, = res.fetchone()
    return {'count':result}
```

URL som returnerer JSON data, skal tolkes av JS-kode på klientsiden

Flask tolker dict som JSON og sender passende HTTP headers

AJAX - pattern
client side updates

```
[...]
   <a href="/actor/{{a_id}}"
       onmouseover="fill_tip_text({{a_id}})"
    >
       {{ aname }}
       <span id="tip{{a_id}}" />
  </a>
[...]
```

tom element
som skal fylles ut med DB data

```
<script>
async function fill_tip_text(id) {
  const url = "/api/moviecount/"+id;
  const tooltip = document.getElementById("tip"+id);

  const response = await fetch(url);

   const result = await response.json();
   const count = result["count"];

   tooltip.textContent="acted in "+count+" movies";
  }
</script>
```

Javascript funksjon som aktiveres
onmouseover

fetch() henter JSON data fra APIen

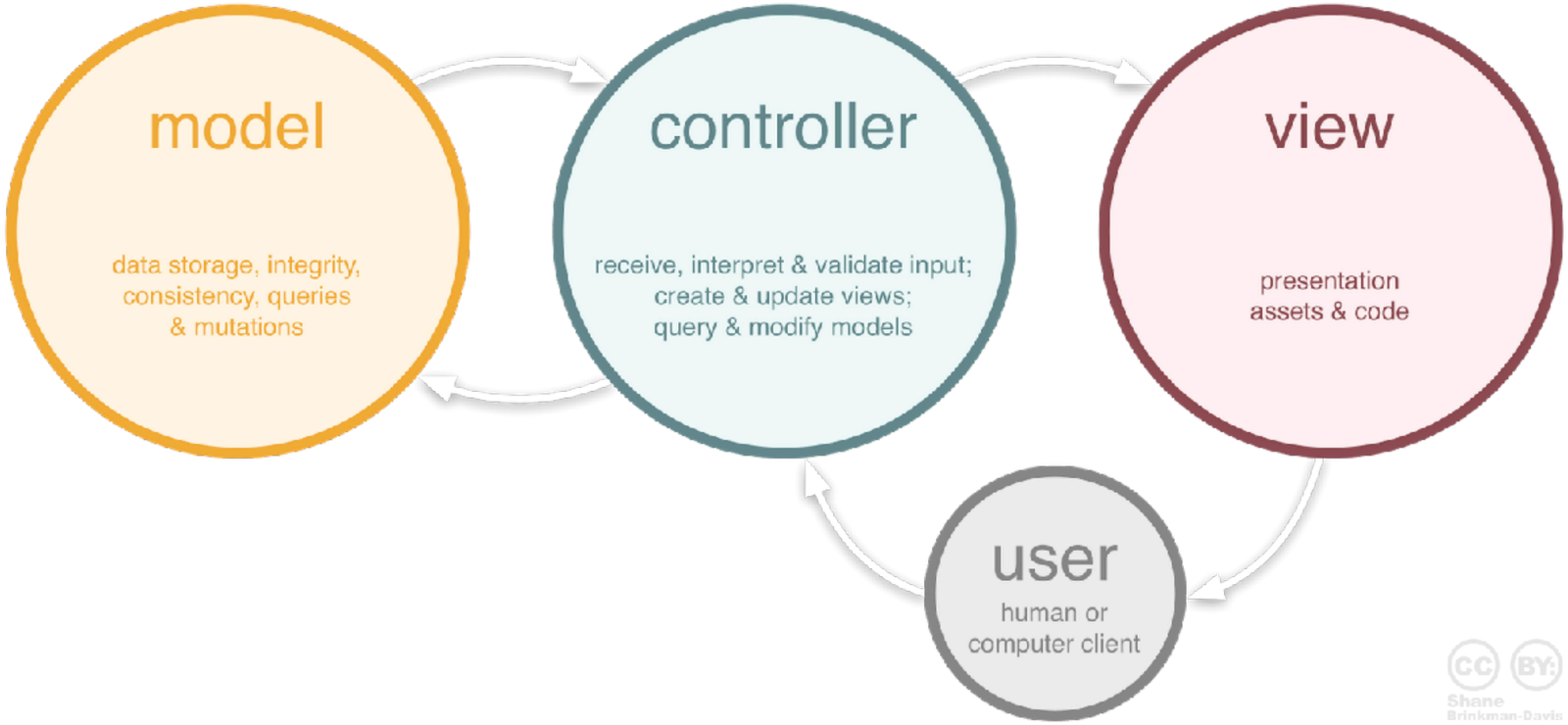
fyller ut tooltip-tekst

- AJAX (Asynchronous JavaScript and XML)
  - started som ActiveX-komponente (Microsoft)
  - implementert som “XMLHttpRequest” JavaScript-objekt
    - asynkrone kall til en URL som forventer XML data tilbake (flest nå JSON), kan brukes til å oppdatere en (små) del av en side uten full reload
    - XMLHttpRequest nå erstattet med fetch()
  - først brukt i Gmail 2004 og Google Maps i 2005
  - AJAX-design sammen med HTML og CSS tillater web-applikasjoner som kan se ut og fungere helt som native apps

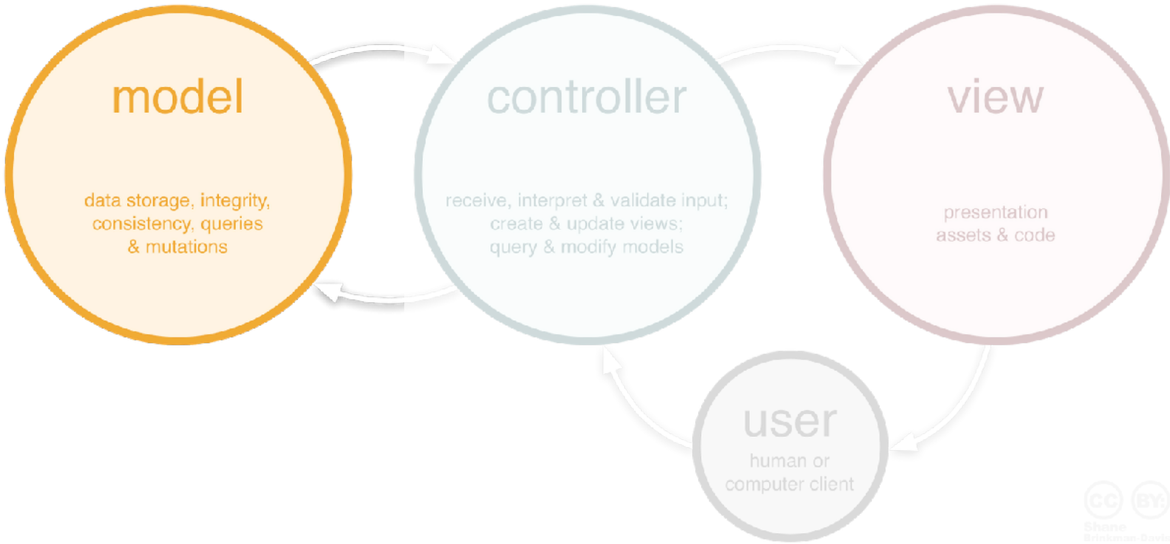
MVC - designmønsteret

- *Model - View - Controller* er en måte å organisere koden bak en nettside
- å følge MVC hjelper med å holde web-applikasjoner ryddig
- klar fordeling av oppgaver
- tillater isolerte lokale endringer uten å påvirke hele appen
  - eksempel: movies lagret som json filer -> som sqlite DB

MVC designmønster
Model — View — Controller



MVC designmønster
Model — View — Controller



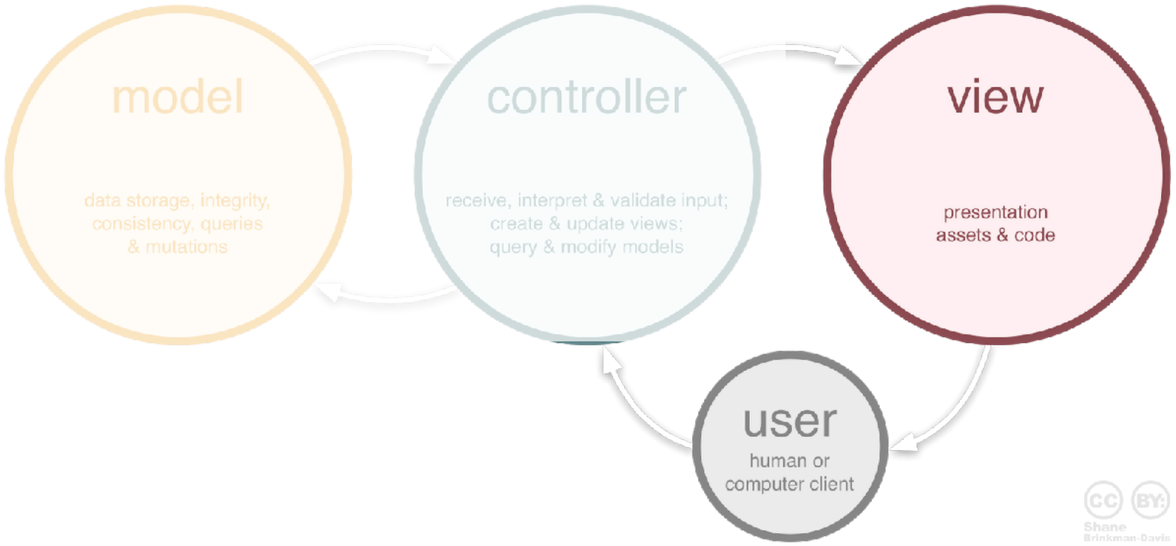
Model:

kommuniserer med databasen: insert, update, select

ofte én OO-klasse til hver DB-tabell

gjenspeiler DB-design og relasjoner

MVC designmønster
Model — View — Controller



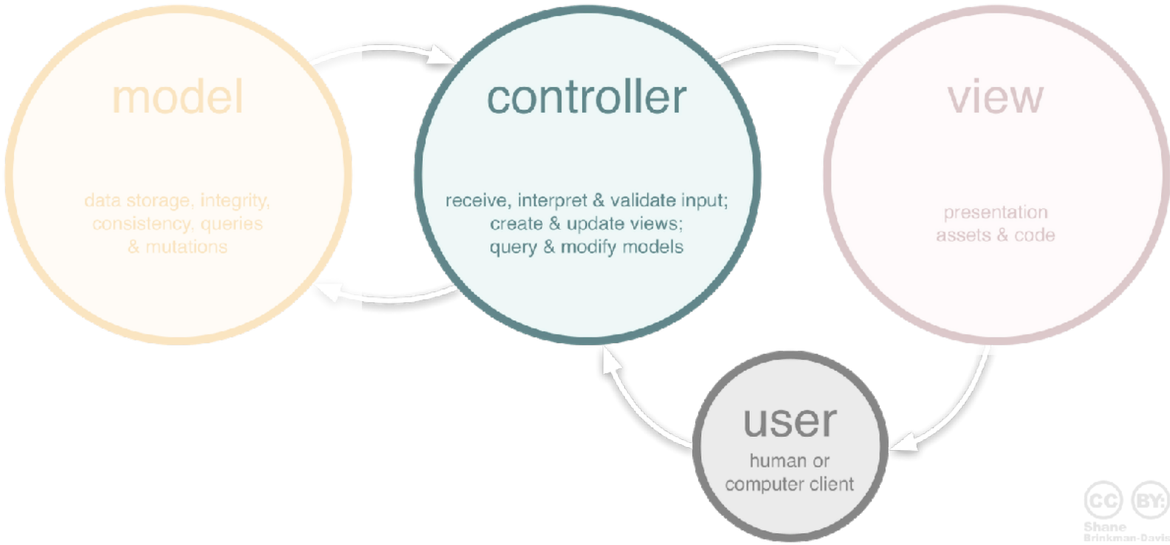
View:

viser informasjon til brukeren gjennom text, bilder, diagrammer og presenterer knapper / søkefelt / ... til interaskjon

en *view* representerer et lite utvalg av DB-innholdet

f.eks. en HTML-nettside på finn

MVC designmønster
Model — View — Controller



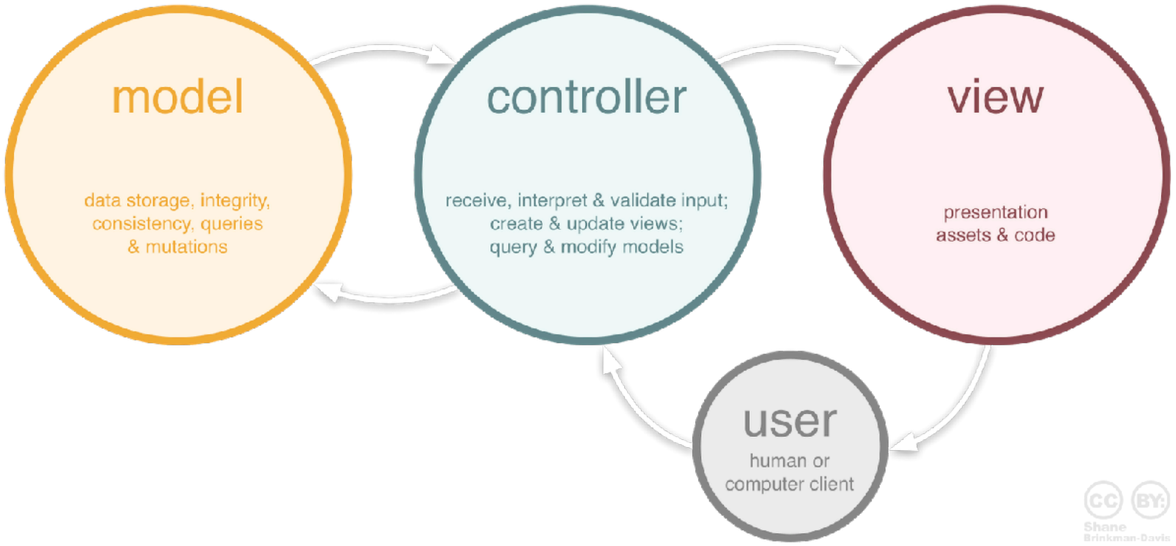
Controller:

tolker bruker-interaskjon, oppdaterer *Model* og/eller *View*

flest av "business logic" skjer her

f.eks: bruker klikket på "kjøp"
-> antall varer i DB må oppdateres,
-> vise bekreftelse og betalingsinfo

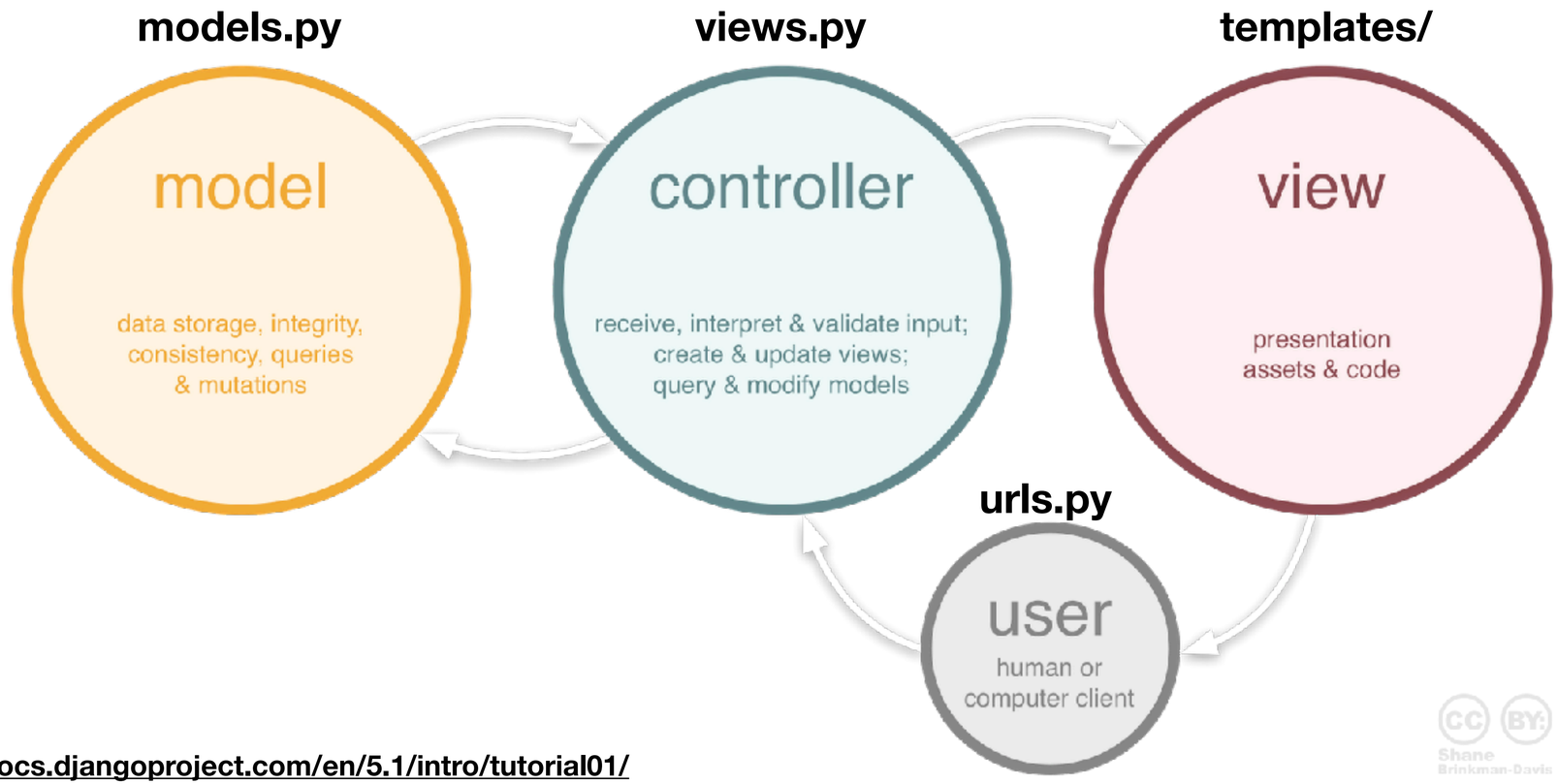
MVC designmønster
Model — View — Controller



De fleste interaktive nettsidene bruker MVC-prinsippet med en DB i bakgrunnen

MVC designmønster
Model — View — Controller

In Django https://www.djangoproject.com/ :



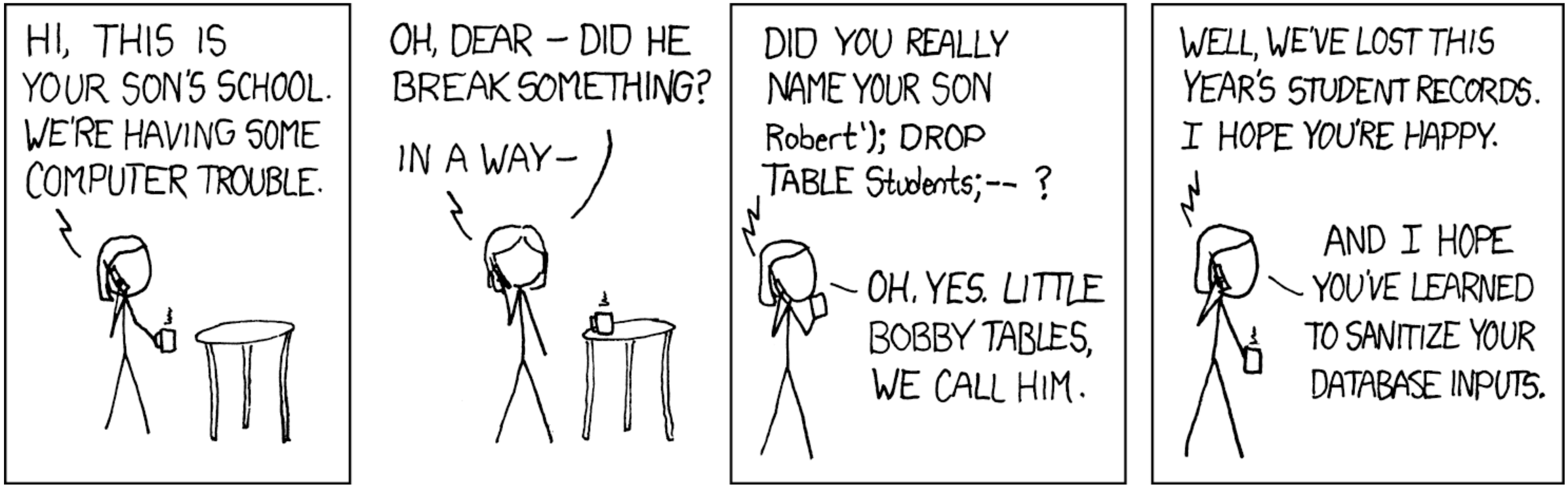
https://docs.djangoproject.com/en/5.1/intro/tutorial01/

Obs når du tolker bruker-input!

SQL injection

- input fra utsiden til bruk i en spørring må sjekkes!
- uid = (...hentet fra brukeren ...)
- query = f"SELECT * FROM customers WHERE USER='{uid}';"
- db.execute(query)

SQL injection



https://xkcd.com/327/

Company forced to change name that could be used to hack websites

Software firm's director thought name using HTML would be 'fun and playful'

The original name of the company was ""><SCRIPT SRC=HTTPS://MJT.XSS.HT> LTD".

Similar names have been registered in the past, such as "; DROP TABLE "COMPANIES";-- LTD"

The British government is changing Companies House name rules to prevent people from creating company names which include code that could sabotage unwary data processors through cross-site scripting (XSS) injection attacks.

https://www.thestack.technology/companies-house-names-rules-drop-table/

https://www.theguardian.com/uk-news/2020/nov/06/companies-house-forces-business-name-change-to-prevent-security-risk

SQL injection

- input fra utsiden til bruk i en spørring må sjekkes!
- uid = ...
  query = f"SELECT * FROM secrets WHERE USER='{uid}';"
- uid = "' OR 1=1; --"
- uid = "' OR ''='"
- uid = "'; DROP TABLE secrets"

SQL injection

- Alltid bruk SQL-biblioteket sin metode til parametre,
  la biblioteket sjekke:

- sqlite3.execute("SELECT * FROM secrets WHERE USER=?", (uid,))

noen sysadmin tasks

- install
- configure services, users
- maintenance, follow logs
- (security) updates
- backups

Hva kjører på maskinen?

- Hvilke prosesser kjører
  ps ax / pstree / top / htop

- hvilke ports er åpen?
  netstat -tulpn / ss -tulpn

- starte / stoppe services (daemons) manuelt
  rc-service nginx restart

- starte services automatisk on boot
  rc-update add nginx default

Alpine bruker openrc -> rc-update osv.

hvert system har sin måte å kontrollere services
f.eks systemctl fra systemd

Hvor finner du log filer?

- Stort sett i /var/log
- kan følge med live
  tail -f /var/log/messages (ctrl-c stopper)
- kernel boot log: dmesg
- web servere som nginx har access log og error log
- det finnes verktøy for å analysere web server logs, hvor mange requests, hvor fra, osv...

Sikkerhet

- Bruk separate vanlige brukere uten doas / sudo - rettigheter til ulike oppgaver
- hvem har tilgang / nøkler til systemet?
- steng services som ikke er i bruk
- bare åpne inngående ports som trengs, les om firewall-settings
- vær kjent med det som skjer på maskinen, hvordan ser en vanlig logfil ut
- oppdater installerte pakker jevnt: apk upgrade

Backups

- Ha en løsning for backups, sjekk at backups virker!
- 3-2-1: 3 copies, on 2 storage types, 1 copy offsite
- ha en rotasjon med gamle backups, f.eks:
  daglig de siste 2 ukene
  ukentlig i 3 måneder
  månedlig i 1 år
  årlig permanent
- ved små systemer er det enklest å ha full backup,
  ellers ha separat strategi til ulike mapper
- *borgbackup* er lett å bruke
- obs på *bitrot*: gamle backups må sjekkes!
  ubrukte USB / SSD mister data ganske raskt sammenlignet med HDD / tape

En fil som er synced på Google Drive
/ Onedrive / Dropbox er IKKE en backup!

Historie: cspring

```
#include <vector>
```

…

```
In file included from foo.cc:1:
…/include/c++/v1/vector:283:10: fatal error:
      'cspring' file not found
#include <cspring>
```

All other copies of <vector>:

```
#include <cstring>
```

cstring -> cspring

ASCII hex codes

00	nul	01	soh	02	stx	03	etx	04	eot	05	enq	06	ack	07	bel
08	bs	09	ht	0a	nl	0b	vt	0c	np	0d	cr	0e	so	0f	si
10	dle	11	dc1	12	dc2	13	dc3	14	dc4	15	nak	16	syn	17	etb
18	can	19	em	1a	sub	1b	esc	1c	fs	1d	gs	1e	rs	1f	us
20	sp	21	!	22	"	23	#	24	$	25	%	26	&	27	'
28	(	29	)	2a	*	2b	+	2c	,	2d	-	2e	.	2f	/
30	0	31	1	32	2	33	3	34	4	35	5	36	6	37	7
38	8	39	9	3a	:	3b	;	3c	<	3d	=	3e	>	3f	?
40	@	41	A	42	B	43	C	44	D	45	E	46	F	47	G
48	H	49	I	4a	J	4b	K	4c	L	4d	M	4e	N	4f	O
50	P	51	Q	52	R	53	S	54	T	55	U	56	V	57	W
58	X	59	Y	5a	Z	5b	[	5c	\	5d	]	5e	^	5f	_
60	`	61	a	62	b	63	c	64	d	65	e	66	f	67	g
68	h	69	i	6a	j	6b	k	6c	l	6d	m	6e	n	6f	o
70	p	71	q	72	r	73	s	74	t	75	u	76	v	77	w
78	x	79	y	7a	z	7b	{	7c	|	7d	}	7e	~	7f	del

01110100 -> 01110000

t -> p

- https://indico.cern.ch/event/13797/contributions/1362288/attachments/115080/163419/Data_integrity_v3.pdf
- On 3000 nodes: write/read 2GB every 2 hours for 5 weeks
- observed single or double bit flips: 1 in 100M, ~10^{-8}
- modern 1TB disk: 80000 bit flips?

ASCII hex codes

00	nul	01	soh	02	stx	03	etx	04	eot	05	enq	06	ack	07	bel
08	bs	09	ht	0a	nl	0b	vt	0c	np	0d	cr	0e	so	0f	si
10	dle	11	dc1	12	dc2	13	dc3	14	dc4	15	nak	16	syn	17	etb
18	can	19	em	1a	sub	1b	esc	1c	fs	1d	gs	1e	rs	1f	us
20	sp	21	!	22	"	23	#	24	$	25	%	26	&	27	'
28	(	29	)	2a	*	2b	+	2c	,	2d	-	2e	.	2f	/
30	0	31	1	32	2	33	3	34	4	35	5	36	6	37	7
38	8	39	9	3a	:	3b	;	3c	<	3d	=	3e	>	3f	?
40	@	41	A	42	B	43	C	44	D	45	E	46	F	47	G
48	H	49	I	4a	J	4b	K	4c	L	4d	M	4e	N	4f	O
50	P	51	Q	52	R	53	S	54	T	55	U	56	V	57	W
58	X	59	Y	5a	Z	5b	[	5c	\	5d	]	5e	^	5f	_
60	`	61	a	62	b	63	c	64	d	65	e	66	f	67	g
68	h	69	i	6a	j	6b	k	6c	l	6d	m	6e	n	6f	o
70	p	71	q	72	r	73	s	74	t	75	u	76	v	77	w
78	x	79	y	7a	z	7b	{	7c	|	7d	}	7e	~	7f	del

<complex>

```
template<class _Xp> _LIBCPP_INLINE_VISIBILITY complex& operator+=(const complex<_Xp>& __c)
    {
        __re_ += __c.real();
        __im_ += __c.imag();
        return *this;
    }
```

Bit flip here would be **very** hard to find

noen sysadmin tasks

- install
- configure services, users
- maintenance, follow logs
- access, uptime
- (security) updates
- backups

Timeplan

- én gruppetime igjen: torsdag / mandag - med Quiz 4 og prosjektstart
- én forelesning igjen: på **onsdag**

- ellers frem til 30.11. :

  drop-in i alle andre gruppe- og forelesningstimene i planen
  (rommene er uendret)

- Kom i tiden! Vi går hjem hvis rommet er tom :-)