Engineer

# Cachning

⏭ att spara ett svar så att arbetet inte görs om

## Vad det är

Cachning är praktiken att behålla resultatet av något som redan har räknats fram, som ett HTTP-svar, svaret på en databasfråga eller en renderad sida, och lämna ut det till senare begäranden enligt regler som säger hur länge resultatet går att lita på och vad som gör det inaktuellt. Verbet är att cacha, och ett resultat som sparats på det sättet är cachat. När ett program sparar returvärdena från en enda funktion internt kallas det oftast memoisering.

Lagret som håller kopiorna är cachen. Cachning är reglerna runt den, och reglerna ska besvara tre frågor. Hur länge är en kopia färsk, alltså duglig att servera utan att fråga någon? Hur kontrolleras den när den har blivit inaktuell? Och vad tar bort eller ersätter kopian när originalet ändras?

HTTP besvarar alla tre i svarshuvudena. RFC 9111 kallar ett lagrat svar färskt så länge dess ålder är lägre än dess färskhetslivstid, som ursprungsservern sätter med direktivet `max-age` i `Cache-Control` eller med ett datum i `Expires`. Därefter är svaret inaktuellt (stale), och en cache måste normalt validera det innan det återanvänds. Den skickar då en villkorlig begäran med en validerare som den har sparat, oftast svarets `ETag` i huvudet `If-None-Match`, och om ingenting har ändrats svarar ursprungsservern 304 Not Modified utan någon kropp. Kontrollen kostar fortfarande en tur och retur, men den sparar överföringen och arbetet med att bygga kroppen en gång till.

```
HTTP/1.1 200 OK
Cache-Control: max-age=3600, stale-while-revalidate=60
ETag: "v42"
```

Inne i en applikation tar samma frågor andra former. Med cache-aside läser koden cachen först, och vid en miss läser den databasen och fyller själv på cachen. Med write-through uppdaterar varje skrivning cachen tillsammans med databasen, så att en läsning hittar det aktuella värdet. En utgångstid på varje post, dess time to live eller TTL, gör samma jobb som `max-age`. Även en databas kan cacha: en materialiserad vy i PostgreSQL lagrar resultatet av en fråga som en tabell, och det resultatet ändras först när någon kör `REFRESH MATERIALIZED VIEW`.

Använd begreppet i ett beslut

Använd ”Vad är cachning?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Hittar databasfrågorna och de saknade indexen bakom ett system som har blivit tyngre för varje år, och åtgärdar dem som spelar roll.

**Se våra tjänster** Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. 30 dagars eldata på 180 ms — 180 ms för att aggregera en månads mätaravläsningar

**30 dagars eldata på 180 ms** Hör av er →

## Varför det spelar roll

Varje regel för cachning byter färskhet mot fart, och rätt byte beror på vilken sorts data det gäller. En logotyp får vara ett år gammal. Ett produktpris får vara en minut gammalt. Ett saldo eller antalet lediga platser bör bekräftas mot källan innan någon agerar på det. Oftast går cachning snett när en och samma regel får gälla för alla tre.

Statiska filer får det renaste svaret. När bygget skriver in en hash av varje fils innehåll i filnamnet får en ändrad fil en ny URL, och den gamla URL:en kan cachas i ett år. RFC 8246 lägger till direktivet `immutable`, som talar om för webbläsaren att svaret förblir detsamma så länge det är färskt, så att den kan hoppa över kontrollen även vid en omladdning. HTML-sidan som namnger filerna måste däremot ha kort livstid eller kontrolleras vid varje användning, annars fortsätter besökarna att ladda de gamla namnen.

Två direktiv är lätta att blanda ihop. `no-cache` låter en cache lagra svaret men förbjuder att det återanvänds innan ursprungsservern har bekräftat det; `no-store` förbjuder att det lagras över huvud taget. För en sida som ändras ofta ger `no-cache` tillsammans med en `ETag` aktuellt innehåll och sparar ändå överföringen när ingenting har ändrats.

Invalidering är den svåra halvan. RFC 9111 kräver att en cache invaliderar en URL när en osäker begäran till den, som `POST`, `PUT` eller `DELETE`, lyckas, och samma avsnitt påpekar att det bara når de cachar som begäran passerar. En webbläsare, ett CDN och en applikationscache är tre skilda platser, och att ta bort en kopia från en av dem lämnar de andra två orörda. Därför låter de flesta system utgångstiden vara skyddsnätet och använder uttrycklig invalidering som ett sätt att bli aktuella tidigare.

Ett inaktuellt svar kan också vara till nytta. RFC 5861 definierar `stale-while-revalidate`, som låter en cache servera ett inaktuellt svar under ett angivet antal sekunder medan den hämtar ett färskt i bakgrunden, och `stale-if-error`, som låter den fortsätta servera en inaktuell kopia när ursprungsservern svarar med ett fel. Båda gör en hård utgångstid mjuk: besökaren får ett svar direkt, och uppdateringen kommer efter.

## Så ser det ut i praktiken

Reglerna för cachning sätts här i var sin ände av ett system, ute på användarens enhet och inne i databasen.

Det ena är en plattform där många av användarna är till sjöss: sjöfolk och fältpersonal, på mobilen och ofta offline. Där gjorde cachningen webbappen installerbar och användbar till sjöss. En service worker sätts upp genom pluginet next-pwa, och körtidscachningen i Workbox serverar typsnitt, bilder, ljud, video och CDN-resurser med cache first: från enheten när en kopia finns där, från nätverket bara vid en miss. Det är filerna som inte ändras från en begäran till nästa, och sidan kombinerar dem med HTTP-huvuden för cachning. Eftersom service workers kräver HTTPS testades installation och offlinebeteende över HTTPS på riktiga enheter. Resultatet som sidan redovisar är en app som går att installera, fortsätter att fungera offline och laddar upprepade besök från cachen i stället för över nätet. En cache first-regel hoppar helt över frågan om färskhet så länge en post finns, och därför passar den filer vars innehåll är knutet till deras adress.

Det andra är en PostgreSQL-databas som blev tio gånger snabbare och som håller användare, tenants, geografiska data och dagliga mätvärden för el. Cachning var en åtgärd bland flera. Vid sidan av riktade index, GiST för spatiala frågor och sammansatta index för filter på tenant och datum, och tidsbaserad partitionering av tabellen med eldata, sparades vissa frågeresultat som materialiserade vyer, och cachning av databasfrågor och anslutningspoolning

sattes upp för att minska overhead. Sidan redovisar 10 gånger bättre svarstider, kritiska operationer som körs på millisekunder och 40 % lägre resursanvändning. Den tillskriver resultatet arbetet som helhet, så hur stor del cachningen stod för är inte mätt för sig.

## Var det går fel

**Efter en driftsättning får en del besökare fortfarande den gamla versionen.** Svaret bar ingen uttrycklig färskhet, så cacharna uppskattade en själva. RFC 9111 låter dem använda en bråkdel av tiden sedan `Last-Modified` och nämner 10 % som en typisk inställning, så en fil som senast ändrades för en månad sedan kan återanvändas i ungefär tre dagar utan att någon frågar. Skicka en uttrycklig `Cache-Control` med varje svar, lägg en innehållshash i namnen på statiska filer och ge HTML-sidan som pekar på dem kort livstid.

**En rensning tömmer CDN:et, men webbläsarna behåller den gamla kopian.** En rensning (purge) tar bort CDN:ets kopia och ingenting annat; varje webbläsare som har lagrat svaret behåller det tills dess egen livstid har löpt ut. Ge publika svar ett kort `max-age` för webbläsare och ett längre `s-maxage`, som bara delade cachar tillämpar, så att den kopia som går att rensa är den som lever längst. När en kopia i webbläsaren måste bytas på en gång, byt URL.

**Ett 404-svar lever kvar efter att sidan har publicerats.** RFC 9110 gör 404 heuristiskt cachebart, så en cache får behålla ett sådant svar som kom utan information om färskhet under en livstid som den själv uppskattar. En länk som följdes en stund innan sidan publicerades kan fortsätta att ge 404 när sidan väl finns. Ge felsvar en uttrycklig kort livstid, och rensa URL:en när sidan går live.

**Ett gammalt värde skrivs tillbaka efter en uppdatering.** Med cache-aside missar en läsare och läser raden; innan den hinner lagra värdet uppdaterar en skrivare raden och tar bort posten; sedan lagrar läsaren det den läste, och det gamla värdet ligger kvar tills något tar bort det. Ta bort posten först när transaktionen har gått igenom med `COMMIT`, och ge varje post en TTL, så att ett värde som blev kvar efter en kapplöpning eller en missad invalidering försvinner av sig självt. I Redis kastar ett vanligt `SET` en tidigare utgångstid, så sätt TTL:en i samma kommando:

```
SET price:42 "19.90" EX 300
```

# Läs mer

- RFC 9111, Section 3: Storing Responses in Caches — när ett svar får lagras, följt av reglerna för färskhet, kontroll mot ursprungsservern och invalidering.
- RFC 9110, Section 13: Conditional Requests — `ETag`, `If-None-Match` och svaret 304, som gör det billigt att kontrollera en kopia på nytt.
- RFC 5861: HTTP Cache-Control Extensions for Stale Content — de två direktiven som låter en cache servera inaktuella svar med avsikt.
- RFC 8246: HTTP Immutable Responses — direktivet `immutable` för filer vars URL ändras med innehållet.
- Workbox: caching strategies — cache first, network first och stale-while-revalidate i en service worker, med kod.
- PostgreSQL: Materialized Views — att hålla ett frågeresultat inuti databasen och uppdatera det på begäran.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med ett representativt dataset, frågorna det ska besvara och dagens flaskhals. Vi kan avtala mått på kvalitet, genomströmning och återställning.

**Se vad det kostar** Hör av er →

## Gör begreppen praktiska

Fråga hur ”Vad är cachning?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

**Se våra tjänster**

welcome@engineer.company +45 50 10 61 84 Köpenhamn, Danmark

Ta med dig den här sidan

Visitkort

**Omsorgsfullt skapad med kärlek av D.**

**CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N**

**Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.**

**Utskrivet från https://engineer.company/sv/glossary/caching/**