Engineer

Bibliotek

kode pakket for at andre programmer skal kalle den

Hva det er

Et bibliotek, eller programbibliotek, er en samling ferdige rutiner som et program låner i stedet for å skrive dem selv: noen løste et problem én gang, som å tolke datoer, komprimere filer eller tegne en QR-kode, og publiserte løsningen med et sett navngitte inngangspunkter som ethvert program kan bruke. C-biblioteker på Linux har ofte navn med prefikset lib-, Windows kaller den delte varianten en DLL, og Rust-programmerere sier crate, et ord som Rust-boken påpeker brukes om hverandre med bibliotek. De fleste språk leveres også med et standardbibliotek, modulene som finnes i hver installasjon uten at noe må legges til.

Et bibliotek er en form for gjenbrukbar kode, og det er lettest å definere mot naboene sine. En applikasjon kjører av seg selv og har en bruker; et bibliotek har ingen av delene og kjører bare inne i et program som kaller det. Et rammeverk er den nærmeste sammenligningen. Med et bibliotek er det koden din som styrer, og den kaller biblioteket når den trenger noe; et rammeverk styrer selv og kaller koden din på de punktene det har definert. En programvarepakke, til slutt, er esken et bibliotek leveres i, med et navn, en versjon og en liste over hva det trenger, og biblioteket er koden inne i esken.

Det som gjør et bibliotek brukbart, er grensesnittet: funksjonene, typene og konstantene det lover å beholde. Semantic Versioning bygger hele ordningen sin på det løftet. Et prosjekt som følger den, må deklarere et offentlig API, og hovedversjonen går opp hver gang det API-et endres på en måte som ikke er bakoverkompatibel.

Program Library HOWTO sorterer biblioteker på Linux etter når de blir en del av programmet. Et statisk bibliotek kopieres inn i den kjørbare filen når den bygges. Et delt bibliotek lastes inn når programmet starter, og brukes av alle programmer

som trenger det, så det kan oppdateres uten at de bygges på nytt. Et dynamisk lastet bibliotek åpnes mens programmet kjører, og det er slik systemer for programtillegg som regel virker.

Bruk begrepet i en beslutning

Bruk «Hva er et bibliotek?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Formen på et system tegnet opp på papiret først, slik at delene passer sammen og kostnaden er kjent før noen skriver kode.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Én infrastrukturplan for 14 avdelinger — 14 forskningsavdelinger på én felles infrastrukturplan

Én infrastrukturplan for 14 avdelinger Ta kontakt →

Hvorfor det er viktig

Hvert bibliotek et prosjekt tar i bruk, er en avgjørelse med tre mulige svar: ta et eksisterende, skrive koden selv, eller kopiere en liten, avgrenset bit inn i repoet og eie den fra da av, noe som kalles vendoring.

Å ta inn et bibliotek er som regel riktig. Forfatterne har allerede møtt grensetilfellene, og andre brukere har allerede meldt inn feilene. Kostnaden kommer senere og er lett å overse: bibliotekets egne avhengigheter følger med, sikkerhetsvarslene for det blir dine, utgivelsesplanen bestemmer takten i oppgraderingene dine, og lisensen blir med inn i produktet ditt. Et bibliotek som vedlikeholderen har forlatt, blir liggende i byggingen din til noen erstatter det.

Å skrive det selv gir kontroll og koster alt biblioteket ville ha spart deg for: grensetilfellene, testene og vedlikeholdet, på ubestemt tid. Det er riktig valg når problemet er smalt og sentralt, når de tilgjengelige bibliotekene er langt større enn behovet, eller når et prosjekt har en grunn til å holde antallet avhengigheter lavt.

Hvordan et bibliotek lenkes inn, betyr også noe. Et program bygd med et statisk bibliotek bærer med seg en fast kopi, så en sikkerhetsrettelse i biblioteket når brukerne først når programmet bygges og leveres på nytt. Et delt bibliotek kan rettes én gang for hele systemet, mot at programmet stoler på den versjonen systemet til enhver tid leverer. Lisensieringen bryr seg ikke om forskjellen: GNU-lisensenes FAQ slår fast at å lenke et GPL-dekket verk statisk eller dynamisk med andre moduler gir et kombinert verk, og at vilkårene i GPL da dekker hele kombinasjonen.

For den som bestiller programvare, er de praktiske spørsmålene hvilke biblioteker resultatet er avhengig av, under hvilke lisenser, og hvem som eier biblioteker leverandøren tar med seg.

Slik ser det ut i praksis

En dokumentgenerator trengte en QR-kode som pekte til nettversjonen av hvert dokument. Ethvert av de eksisterende QR-bibliotekene ville ha gjort jobben, men prosjektet hadde bevisst begrenset seg til to avhengigheter under kjøring, en malmotor og en PDF-renderer, en grense valgt for at generatoren fortsatt skulle kunne installeres uten problemer lenge etter utgivelsen. Svaret ble å vendore en QR-koder på 522 linjer: koding i byte-modus, Reed-Solomon-feilretting over GF(256), posisjons-, tids- og justeringsmønstrene, format- og versjonsinformasjon og alle åtte maskemønstre vurdert mot de fire straffereglene. En dekoder skrevet for seg leser hver genererte kode tilbake ved hver kjøring og sjekker at den gir tilbake inndataene. Siden er nøye med påstanden sin: koderen dekker bare modusene og versjonene disse dokumentene bruker, og grunnen til den er avhengighetsbudsjettet, ikke en tro på at den slår et modent bibliotek.

En plattformtilpasset macOS-klient kaller et kjernebibliotek gjennom C-grensesnittet dets, og tidlig ble det grensesnittet representert av en håndskrevet headerfil. Kall til tre funksjoner som for lengst hadde forsvunnet fra kjernen, kompilerte, ble lenket og krasjet så under kjøring, fordi kompilatoren og lenkeren bare noen gang hadde sett den håndskrevne stedfortrederen og aldri kjernens ekte deklarasjoner. Løsningen ble en parser som verifiserer hvert kallsted mot den ekte headerfilen på 7 308 linjer og dens 268 deklarasjoner, og som får byggingen til å feile ved ethvert kall eller enhver enum-konstant headerfilen ikke deklarerer. Den sjekker deklarasjoner, så en funksjon som beholder signaturen sin mens betydningen endres, slipper fortsatt gjennom.

Vilkårene for hvert oppdrag avgjør hvem som eier hva. Kode som skrives for en kunde, blir kundens ved full betaling. Biblioteker og generelle verktøy selskapet skrev tidligere, eller skrev for seg selv, er unntaket: eierskapet blir hos selskapet, og kunden får en tidsubegrenset, ugjenkallelig lisens til å bruke dem i det leverte arbeidet. Kode med åpen kildekode fra tredjeparter beholder lisensen forfatterne valgte, og kunden får vite hvilke av de komponentene som havnet i resultatet.

Hvor det går galt

En rutineoppdatering knekker byggingen

En avhengighet går fra 2.3 til 2.4, og kode som kompilerte i går, feiler eller oppfører seg annerledes. Enten har biblioteket endret det offentlige grensesnittet sitt uten å øke hovedversjonen, eller så har prosjektet støttet seg på noe biblioteket aldri lovet, som en intern funksjon eller en udokumentert oppførsel. Lås eksakte versjoner i en låsefil, oppgrader med vilje og med endringsloggen åpen, og kall bare det biblioteket dokumenterer som offentlig.

Programmet stoler på en beskrivelse av biblioteket

En ren bygging krasjer første gang en funksjon brukes. Koden ble kompilert mot headerfiler, typestubber eller bindinger skrevet for hånd, og de har drevet bort fra biblioteket som faktisk lenkes inn. Generer bindingene fra bibliotekets eget publiserte grensesnitt, eller sjekk hvert kall mot det som en del av byggingen, slik at et avvik feiler der og ikke foran en bruker.

Én liten funksjon drar med seg et stort tre

Én enkelt funksjon trengtes, og biblioteket som leverer den, kom med dusinvis av egne avhengigheter, hver av dem enda et sett med sikkerhetsvarsler, lisenser og oppdateringer. Før du legger til et bibliotek, se på hva det trekker med seg, ikke bare på hvor stort det selv er. Når behovet er smalt og godt spesifisert, kan en liten bit vendoret kode med egne tester være det billigste å eie.

Lisensen følger med koden

Sent i et prosjekt oppdager noen at et bibliotek inne i et produkt med lukket kildekode er under en lisens der vilkårene gjelder hele det kombinerte verket. Ingen sjekket lisensene da biblioteket ble lagt til, og å lenke det inn dynamisk endret ingenting. Sjekk lisensen til hvert bibliotek i det øyeblikket det legges til, hold listen sammen med utgivelsen, og be leverandørene om deres.

Les mer

- Program Library HOWTO: Introduction — hva et programbibliotek er på Linux, og hvordan statiske, delte og dynamisk lastede biblioteker skiller seg fra hverandre.
- The Rust Programming Language: Packages and Crates — bibliotek-crates mot binære crates, og hvordan en pakke holder dem.

- Semantic Versioning 2.0.0 — reglene for versjonsnumre som gjør et biblioteks offentlige API til et løfte.
- The Python Standard Library — et standardbibliotek i full størrelse: alt som følger med hver installasjon av Python.
- Frequently Asked Questions about the GNU Licenses — hva GPL betyr for programmer som lenker inn et dekket bibliotek, statisk eller dynamisk.

Ordliste

Velg ut fra deres faktiske rammer

Beskriv arbeidsflyten, eksisterende grensesnitt og brukerne som trenger endringen. Vi kan definere en testbar leveranse og en overlevering som gjør teamet i stand til å vedlikeholde løsningen.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er et bibliotek?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/library/