Engineer

Bibliotek

kode, pakket så andre programmer kan kalde den

Hvad det er

Et bibliotek er en samling færdige rutiner, som et program låner i stedet for at skrive dem selv: nogen har løst en opgave én gang, for eksempel at læse datoer, pakke filer sammen eller tegne en QR-kode, og stillet løsningen til rådighed bag en række navngivne indgange, som ethvert program kan kalde. I flertal hedder det biblioteker, i bestemt form biblioteket, og man møder også de længere former programbibliotek og kodebibliotek. På Linux har C-biblioteker ofte et lib-præfiks i filnavnet, Windows kalder den delte slags en DLL, og i Rust siger man crate, et ord, som Rust-bogen skriver bruges i flæng med bibliotek. De fleste sprog leveres desuden med et standardbibliotek: de moduler, der findes i enhver installation, uden at man skal tilføje noget.

Et bibliotek er genbrugelig kode, og det er lettest at forklare ud fra naboerne. En applikation kører af sig selv og har en bruger; et bibliotek har ingen af delene og kører kun inde i et program, der kalder det. Et framework ligger tættere på. Med et bibliotek er det programmets egen kode, der styrer og kalder biblioteket, når der er brug for det; et framework styrer selv og kalder programmets kode de steder, det har gjort plads til. En pakke er endelig den kasse, et bibliotek sendes ud i, med navn, version og en liste over, hvad det kræver, mens biblioteket er koden i kassen.

Det, der gør et bibliotek brugbart, er dets grænseflade: de funktioner, typer og konstanter, det lover at holde fast i. Semantic Versioning bygger hele sit system på det løfte. Et projekt, der følger det, skal erklære et offentligt API, og hovedversionen stiger, hver gang det API ændres på en måde, der bryder med det gamle.

Program Library HOWTO inddeler biblioteker på Linux efter, hvornår de kommer ind i programmet. Et statisk bibliotek kopieres ind i den eksekverbare fil, når den bygges. Et delt bibliotek indlæses, når programmet starter, og deles af alle programmer, der bruger det, så det kan opdateres, uden at de skal bygges igen. Et dynamisk indlæst bibliotek åbnes, mens programmet kører, og det er sådan, plugin-systemer som regel virker.

Brug begrebet i en beslutning

Brug »Hvad er et bibliotek?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Systemets form lagt fast på papir først, så delene passer sammen, og prisen er kendt, før nogen skriver kode.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Én infrastrukturplan til 14 afdelinger — 14 forskningsafdelinger på én infrastrukturplan

Én infrastrukturplan til 14 afdelinger Kontakt os →

Hvorfor det betyder noget

Hvert bibliotek, et projekt tager ind, er en beslutning med tre mulige svar: brug et eksisterende, skriv koden selv, eller kopiér et lille, afgrænset stykke ind i repositoriet og ej det fra da af. Det sidste kaldes vendoring.

Et eksisterende bibliotek er som regel det rigtige valg. Forfatterne har allerede mødt grænsetilfældene, og andre brugere har meldt fejlene. Regningen kommer senere og er let at overse: bibliotekets egne afhængigheder følger med, dets sikkerhedsadvarsler bliver projektets, dets udgivelsesrytme bestemmer tempoet i opgraderingerne, og dets licens rejser med ind i produktet. Et bibliotek, som vedligeholderen har opgivet, bliver i buildet, indtil nogen skifter det ud.

At skrive koden selv giver kontrol og koster alt det, biblioteket ville have sparet: grænsetilfældene, testene og vedligeholdelsen, så længe koden lever. Det er det rigtige valg, når problemet er smalt og centralt, når de eksisterende biblioteker er langt større end behovet, eller når projektet har en grund til at holde antallet af afhængigheder nede.

Måden, et bibliotek linkes på, spiller også ind. Et program bygget med et statisk bibliotek bærer en fast kopi, så en sikkerhedsrettelse i biblioteket først når brugerne, når programmet bygges og udgives igen. Et delt bibliotek kan rettes én gang for hele systemet, til gengæld stoler programmet på den version,

systemet leverer. Licenserne er ligeglade med forskellen: GNU's licens-FAQ slår fast, at et GPL-dækket værk, der linkes statisk eller dynamisk med andre moduler, udgør et samlet værk, og så dækker GPL'ens vilkår hele kombinationen.

For den, der bestiller software, er de praktiske spørgsmål, hvilke biblioteker resultatet afhænger af, under hvilke licenser, og hvem der ejer de biblioteker, leverandøren selv tager med.

Sådan ser det ud i praksis

En dokumentgenerator skulle sætte en QR-kode på hvert dokument med et link til netudgaven. Ethvert af de eksisterende QR-biblioteker ville have klaret opgaven, men projektet havde med vilje holdt sig til to afhængigheder ved kørsel, en skabelonmotor og en PDF-renderer, for at generatoren også ville lade sig installere om mange år. I stedet fik projektet sin egen QR-encoder på 522 linjer: kodning i byte-tilstand, Reed-Solomon-fejlkorrektion over GF(256), positions-, timing- og justeringsmønstrene, format- og versionsinformation og alle otte maskemønstre, vurderet efter de fire strafregler. En dekoder, der er skrevet uafhængigt af encoderen, læser hver genereret kode tilbage ved hver kørsel og kontrollerer, at resultatet er det samme som inputtet. Siden er omhyggelig med sin påstand: encoderen dækker kun de tilstande og versioner, dokumenterne bruger, og begrundelsen er budgettet for afhængigheder, ikke en tro på, at den slår et modent bibliotek.

En native macOS-klient kalder et kernebibliotek gennem dets C-grænseflade, og i starten var den grænseflade beskrevet i en håndskrevet header. Kald til tre funktioner, der var forsvundet fra kernen, blev kompileret og linket og gik så ned under kørslen, fordi både compiler og linker kun havde set den håndskrevne stedfortræder og aldrig kernens rigtige erklæringer. Løsningen blev en parser, der efterprøver hvert kaldsted, mod den rigtige header på 7.308 linjer med 268 erklæringer, og den stopper buildet ved ethvert kald og enhver enum-konstant, som headeren ikke erklærer. Den tjekker erklæringer, så en funktion, der beholder sin signatur, men skifter betydning, slipper igennem.

Vilkårene for hvert samarbejde fastlægger, hvem der ejer hvad. Kode, der skrives til en kunde, bliver kundens ved fuld betaling. Biblioteker og generelle værktøjer, som virksomheden har skrevet tidligere eller til sig selv, er undtagelsen: de forbliver virksomhedens, og kunden får en varig licens, der ikke kan trækkes

tilbage, til at bruge dem i det leverede. Open source-kode fra andre følger fortsat den licens, ophavsfolkene har valgt, og kunden får at vide, hvilke af de komponenter der er endt i resultatet.

Hvor det går galt

En rutineopdatering vælter buildet

En afhængighed går fra 2.3 til 2.4, og kode, der kompilerede i går, fejler eller opfører sig anderledes. Enten ændrede biblioteket sin offentlige grænseflade uden at hæve hovedversionen, eller også byggede projektet på noget, biblioteket aldrig har lovet, som en intern funktion eller en udokumenteret adfærd. Lås præcise versioner i en lockfil, opgradér bevidst med ændringsloggen åben, og kald kun det, biblioteket dokumenterer som offentligt.

Programmet stoler på en beskrivelse af biblioteket

Et build, der gik igennem uden en eneste fejl, går ned, første gang en bestemt funktion bliver brugt. Koden blev kompileret mod headere, typestubbe eller bindinger, som nogen skrev i hånden, og de gled væk fra det bibliotek, der rent faktisk bliver linket ind. Generér bindingerne fra bibliotekets egen offentliggjorte grænseflade, eller tjek hvert kald mod den som en del af buildet, så en uoverensstemmelse stopper dér og ikke foran en bruger.

En lille funktion trækker et stort træ med sig

Der var brug for én funktion, og det bibliotek, der leverer den, kom med dusinvis af egne afhængigheder, hver med sine egne sikkerhedsadvarsler, licenser og opdateringer. Se på, hvad et bibliotek trækker med sig, før det tages ind, og ikke kun på dets egen størrelse. Når behovet er smalt og veldefineret, kan et lille stykke kode kopieret ind i repositoriet med sine egne test være det billigste at eje.

Licensen følger med koden

Sent i et projekt opdager nogen, at et bibliotek i et lukket produkt er under en licens, hvis vilkår omfatter hele det samlede værk. Ingen tjekkede licenser, da biblioteket blev tilføjet, og at det blev linket dynamisk, ændrede intet. Tjek licensen på hvert bibliotek i det øjeblik, det tilføjes, gem listen sammen med udgivelsen, og bed leverandører om deres.

Læs mere

- Program Library HOWTO: Introduction — hvad et programbibliotek er på Linux, og hvordan statiske, delte og dynamisk indlæste biblioteker adskiller sig fra hinanden.
- The Rust Programming Language: Packages and Crates — library crates over for binary crates, og hvordan en pakke rummer dem.
- Semantic Versioning 2.0.0 — reglerne for versionsnumre, der gør et biblioteks offentlige API til et løfte.
- The Python Standard Library — et standardbibliotek i fuld størrelse: alt det, der følger med enhver Python-installation.
- Frequently Asked Questions about the GNU Licenses — hvad GPL betyder for programmer, der linker et GPL-dækket bibliotek, statisk eller dynamisk.

Ordliste

Vælg ud fra jeres faktiske vilkår

Beskriv arbejdsgangen, de eksisterende grænseflader og de brugere, der har brug for ændringen. Vi kan definere en testbar leverance og en overdragelse, der gør teamet i stand til at vedligeholde løsningen.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er et bibliotek?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

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

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/library/