Engineer

Bibliotek

kod som paketerats för att andra program ska anropa den

Vad det är

Ett bibliotek är en samling färdiga rutiner som ett program lånar i stället för att skriva dem själv: någon har löst ett problem en gång, t.ex. att tolka datum, komprimera filer eller rita en QR-kod, och gett ut lösningen med en uppsättning namngivna ingångar som vilket program som helst kan använda. När sammanhanget annars kunde handla om böcker säger man programbibliotek. C-bibliotek på Linux har ofta prefixet lib- i namnet, Windows kallar den delade sorten DLL, och den som skriver Rust säger crate, ett ord som Rust-boken uppger används omväxlande med library. De flesta språk levereras dessutom med ett standardbibliotek, de moduler som finns i varje installation utan att något behöver läggas till.

Ett bibliotek är ett slags återanvändbar kod, och det är lättast att definiera mot sina grannar. En applikation körs på egen hand och har en användare; ett bibliotek har ingetdera och körs bara inuti ett program som anropar det. Ett ramverk är den närmare jämförelsen. Med ett bibliotek är det din kod som bestämmer och anropar biblioteket när den behöver något; ett ramverk bestämmer själv och anropar din kod på de ställen som det har definierat. Ett programpaket, slutligen, är lådan som ett bibliotek levereras i, med ett namn, en version och en lista över vad det behöver, och biblioteket är koden i lådan.

Det som gör ett bibliotek användbart är dess gränssnitt: de funktioner, typer och konstanter som det lovar att behålla. Semantic Versioning bygger hela sitt system på det löftet. Ett projekt som följer det måste deklarera ett publikt API, och dess huvudversion räknas upp så snart API:t ändras på ett sätt som inte är bakåtkompatibelt.

Program Library HOWTO delar in biblioteken på Linux efter när de kommer in i programmet. Ett statiskt bibliotek kopieras in i den körbara filen när den byggs. Ett delat bibliotek laddas när programmet startar och används av alla program som behöver det, så att det kan uppdateras utan att de byggs om. Ett dynamiskt laddat bibliotek öppnas medan programmet körs, och det är oftast så system för insticksprogram fungerar.

Använd begreppet i ett beslut

Använd ”Vad är ett bibliotek?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Systemets form utarbetad på papper först, så att delarna passar ihop och kostnaden är känd innan någon skriver kod.

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. En infrastrukturplan för 14 avdelningar — 14 forskningsavdelningar under en och samma infrastrukturplan

En infrastrukturplan för 14 avdelningar Hör av er →

Varför det spelar roll

Varje bibliotek som ett projekt tar in är ett beslut med tre möjliga svar: att ta ett befintligt, att skriva koden i egen regi, eller att kopiera in en liten, avgränsad bit i det egna kodförrådet och äga den från och med då, det som på engelska kallas vendoring.

Att ta ett bibliotek är oftast rätt. Dess upphovspersoner har redan stött på specialfallen, och andra användare har redan rapporterat buggarna. Kostnaden kommer senare och är lätt att missa: bibliotekets egna beroenden följer med, dess säkerhetsbulletiner blir dina, dess utgivningstakt bestämmer takten i dina uppgraderingar, och dess licens följer med in i din produkt. Ett bibliotek som underhållaren har övergett ligger kvar i bygget tills någon byter ut det.

Att skriva koden själv ger kontroll och kostar allt som biblioteket skulle ha sparat: specialfallen, testerna och underhållet, på obestämd tid. Det är rätt val när problemet är smalt och centralt, när de tillgängliga biblioteken är mycket större än behovet, eller när ett projekt har skäl att hålla nere antalet beroenden.

Hur ett bibliotek länkas spelar också in. Ett program som har byggts med ett statiskt bibliotek bär på en fast kopia, så en säkerhetsrättning i biblioteket når användarna först när programmet byggs om och levereras på nytt. Ett delat bibliotek kan rättas en gång för hela systemet, till priset att programmet litar på den version som systemet tillhandahåller. Licensen gör ingen skillnad mellan de

två: GNU:s vanliga frågor om licenserna slår fast att den som länkar ett verk under GPL statiskt eller dynamiskt med andra moduler skapar ett kombinerat verk, och att GPL:s villkor då omfattar hela kombinationen.

För den som beställer programvara är de praktiska frågorna vilka bibliotek resultatet är beroende av, under vilka licenser, och vem som äger de bibliotek som leverantören tar med sig in i arbetet.

Så ser det ut i praktiken

En dokumentgenerator behövde en QR-kod som pekade på varje dokuments onlineversion. Vilket som helst av de befintliga QR-biblioteken hade klarat uppgiften, men projektet hade medvetet hållit sig till två beroenden vid körning, en mallmotor och en PDF-renderare, en gräns som valts för att generatorn fortfarande ska gå att installera om flera år. Svaret blev att bygga in en QR-kodare på 522 rader: kodning i byteläge, felkorrigering med Reed-Solomon över Galoiskroppen med 256 element, positionsmönster, tidsmönster och justeringsmönster, format- och versionsinformation, samt alla åtta datamaskmönster, som utvärderas mot de fyra reglerna för straffpoäng. En avkodare som har skrivits oberoende läser tillbaka varje genererad kod vid varje körning och kontrollerar att den avkodade strängen är lika med indata. Sidan är noga med vad den påstår: kodaren stöder bara de lägen och versioner som dokumenten använder, och skälet till att den skrevs är beroendebudgeten, inte en tro att den är bättre än ett moget bibliotek.

En macOS-applikation skriven i Swift anropar ett kärnbibliotek via dess C-gränssnitt, och tidigt i projektet representerades det gränssnittet av en handskriven headerfil. Anrop till tre funktioner som inte längre fanns i kärnan kompilerade, länkades och kraschade sedan vid körning, eftersom kompilatorn och länkaren bara hade sett den handskrivna platshållaren och aldrig kärnans verkliga deklarationer. Lösningen blev en parser som verifierar varje anropsställe mot den riktiga headerfilen på 7 308 rader och dess 268 deklarationer, och som får bygget att fallera vid varje anrop eller enum-konstant som headerfilen inte deklarerar. Parsern kontrollerar deklarationer, så en funktion som behåller sin signatur men byter betydelse passerar ändå.

Villkoren för varje uppdrag avgör vem som äger vad. Kod som skrivs för en kund blir kundens när full betalning har skett. Bibliotek och allmänna verktyg som företaget skrev före projektet, eller för eget bruk, är undantaget: de förblir företagets, och kunden får en tidsobegränsad, oåterkallelig licens att använda

dem i det levererade arbetet. Komponenter med öppen källkod från tredje part behåller de licenser som deras upphovspersoner valde, och kunden får veta vilka sådana komponenter som ingår i resultatet.

Var det går fel

En rutinmässig uppdatering knäcker bygget

Ett beroende går från version 2.3 till 2.4, och kod som kompilerade i går fallerar eller beter sig annorlunda. Antingen har biblioteket ändrat sitt publika gränssnitt utan att räkna upp huvudversionen, eller så har projektet förlitat sig på något som biblioteket aldrig lovade, t.ex. en intern funktion eller ett odokumenterat beteende. Lås exakta versioner i en låsfil, uppgradera avsiktligt med ändringsloggen öppen, och anropa bara det som biblioteket dokumenterar som publikt.

Programmet litar på en beskrivning av biblioteket

Ett rent bygge kraschar första gången någon använder en viss del av programmet. Koden kompilerades mot headerfiler, typstubbar eller bindningar som skrivits för hand, och de har glidit isär från det bibliotek som faktiskt länkas in. Generera bindningarna från bibliotekets eget publicerade gränssnitt, eller kontrollera varje anrop mot det som ett steg i bygget, så att en avvikelse fallerar där och inte framför en användare.

En liten funktion drar in ett stort träd

Det behövdes en enda funktion, och biblioteket som tillhandahåller den kom med dussintals egna beroenden, vart och ett med sina egna säkerhetsbulletiner, licenser och uppdateringar. Titta på vad ett bibliotek drar in innan du lägger till det, inte bara på dess egen storlek. När behovet är smalt och väl specificerat kan en liten bit kod som kopierats in i kodförrådet med egna tester vara det billigare att äga.

Licensen följer med koden

Sent i ett projekt märker någon att ett bibliotek i en produkt med sluten källkod har en licens vars villkor sträcker sig till hela det kombinerade verket. Licensen kontrollerades aldrig när biblioteket lades till, och att länka det dynamiskt ändrade ingenting. Kontrollera licensen för varje bibliotek i samma stund som det läggs till, spara listan tillsammans med utgåvan, och be leverantörerna om deras.

Läs mer

- Program Library HOWTO: Introduction — vad ett programbibliotek är på Linux, och hur statiska, delade och dynamiskt laddade bibliotek skiljer sig åt.
- The Rust Programming Language: Packages and Crates — bibliotekscrates jämfört med binära crates, och hur ett paket rymmer dem.
- Semantic Versioning 2.0.0 — reglerna för versionsnummer som gör ett biblioteks publika API till ett löfte.
- The Python Standard Library — ett standardbibliotek i full storlek: allt som följer med varje Python-installation.
- Frequently Asked Questions about the GNU Licenses — vad GPL betyder för program som länkar ett bibliotek under licensen, statiskt eller dynamiskt.

Ordlista

Välj utifrån era faktiska förutsättningar

Beskriv arbetsflödet, befintliga gränssnitt och användarna som behöver förändringen. Vi kan definiera en testbar leverans och en överlämning som gör att teamet kan underhålla lösningen.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är ett bibliotek?” 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/library/