

معماری تمیز

Clean Architecture

راهنمای استادی در طراحی و ساختار نرم افزار

ترجمه و تألیف

مهرداد عباسی - سید مرتضی قائم مقامی

سرشناسه: مارتین، رابرت سی. Martin, Robert C.

عنوان و مترجمین: معماری تمیز: راهنمای استادی در طراحی و ساختار نرم افزار / [تالیف: رابرت سی مارتین]؛ ترجمه و تالیف: مهرداد عباسی، سیدمرتضی قائم مقامی.

« مشخصات نشر: اصفهان، موسسه تولید علم فرزندگان برنا، ۱۳۹۸.

« شابک: ۹۷۸-۶۰۰-۸۲۰۲-۸۰-۶ « مشخصات ظاهری: ۴۰۶ص، وزیری

« وضعیت فهرست نویسی: فپا

« یادداشت: عنوان اصلی

» Clean architecture : a craftsman's guide to software structure and design, 2018.

« عنوان دیگر: راه می است: در طراحی و ساختار نرم افزار

« موضوع: معماری نرم رار « موضوع: Software architecture

« موضوع: نرم افزار -- تولید « موضوع: Computer software -- Development

« موضوع: Com pute programming--History.

« شناسه افزوده: عباسی، مهرداد، ۱۳۶۵، م.

« شناسه افزوده: قائم مقامی، سید مرتضی، ۱۳۶۶، م. جم

« رده بندی کنگره: QA ۷۶/۷۶ « رده بندی دیو ی: ۰۰۴-۷۲

« شماره کتابشناسی ملی: ۵۸۹۳۹۰۲

نشانی ناشر: صفار خیابان بزرگمهر، خیابان هشت

بهشت غربی، مجتمع فک، پلاک ۱۰، بقیه سوم، واحد ۳۰۱

تلفن انتشارات: ۴۰۲۶۶-۰۹۱۳

تلفن فروش: ۳۳۷۴۶۷-۰۹۱۳

چاپ اول: ۱۳۹۸

عنوان: معماری تمیز: راهنمای استادی در طراحی و

ساختار نرم افزار

ترجمه و تالیف: مهرداد عباسی - سید مرتضی قائم مقامی

صفحه آرای: زهرا مسائل

طراح جلد: منیژه عباسی

تیراژ: ۵۰۰



Nashrtolidelm@gmail.com

قیمت: ۱۴۰۰۰۰ تومان

* هرگونه چاپ و تکثیر (اعم از زیراکس، بازنویسی و ضبط کامپیوتری) از محتویات

این اثر بدون اجازه کتبی از مترجمین ممنوع است.

متخلفان به موجب بند ۵ از ماده ۲ قانون حمایت از مولفان، مصنفان و هنرمندان تحت

پیگرد قانونی قرار می گیرند.

این کتاب تنها به صورت آنلاین و انحصاراً توسط سایت zerobook.ir به فروش می رسد.

کلیه حقوق مادی و معنوی این اثر متعلق به مترجمین است.

سخن مترجمین

کتاب پیش روی شما ترجمه کتاب Clean Architecture اثر Robert C. Martin (معروف به عموباب) می باشد. این کتاب شامل چند دهه تجربه برنامه نویسی و چگونگی رفتار در پروژه های مختلف برنامه نویسی است و خواندن آن برای ورود به دنیای حرفه ای برنامه نویس ها توصیه می شود. فصل های کتاب شامل موضوعاتی مانند معماری، برنامه نویسی ساخت یافته، برنامه نویسی شی گرا، برنامه نویسی تابعی، اصل SRP، اصل OCP، اصل LSP، اصل DIP، قوانین کسب و کار و سرویس ها است.

خوانندگان عزیز توجه داشته باشند که در راستای فهم و درک بهتر مطالب، برخی مثال ها به گونه ای ترجمه شده است، تا کاربران فارسی زبان راحت تر بتوانند مطلب را درک کنند.

با توجه به دقت فراوانی که در جهت ارائه ترجمه مناسب و قابل فهم برای شما خوانندگان عزیز صورت گرفته است، ممکن است اشکالاتی در متن کوچک نیز وجود داشته باشد. لذا از تمام عزیزان خواهشمندیم که در صورت داشتن هر گونه پیشنهاد یا انتقاد جهت اصلاح یا بهتر شدن این کتاب، از طریق سایت zerobook.ir ما را از نظرات خود مطلع سازید.

در پایان خداوند را سپاسگزاریم که توفیق به انجام رساندن این کار را به ما اعطا کرد و به امید آنکه شما خوانندگان گرامی در سایه الطاف خداوند متعال همواره سلامت و موفق باشید.

مهرداد عباسی

MehردادLinux@gmail.com

سید مرتضی قائم مقامی

mortezaaghaem@gmail.com

فهرست مطالب

شماره صفحه

عنوان

۱ پیشگفتار

۷ مقدمه

۱۱ درباره نویسنده

بخش اول: معرفی

۱۵ فصل ۱: طرازم و معماری چیست؟

۱۶ هدف چیست؟

۱۷ مثال مرتبط

۲۵ نتیجه گیری

۲۷ فصل ۲: داستان دو ارزش

۲۸ عملکرد

۲۸ معماری

۲۹ ارزش مهم تر

۳۰ ماتریس آیزنهاور

۳۲ جنگ بر سر معماری

بخش دوم: شروع سافت و سازه، نحوه ی برنامه نویسی

۳۵ فصل ۳: بررسی اجمالی الگوهای برنامه نویسی

۳۶ برنامه نویسی ساخت یافته

۳۶ برنامه نویسی شی گرا

۳۷ برنامه نویسی تابعی

۳۸ غذای فکر

۳۸ نتیجه گیری

۳۹ فصل ۴: برنامه نویسی سافت یافته

۴۰ برهان

۴۲ یک اعلان زبان آور

فهرست مطالب

شماره صفحه

عنوان

۴۳	تجزیه تابعی
۴۳	هیچ اثباتی رسماً ارائه نشد
۴۴	علم برای رهایی
۴۴	آزمایش‌ها
۴۵	نتیجه‌گیری

۴۷	فصل ۵: برنامه‌نویسی شی‌گرا
۴۸	کپسوله‌سازی؟
۵۱	وارثت؟
۵۳	چندریختی؟
۶۰	نتیجه

۶۱	فصل ۶: برنامه‌نویسی تابعی
۶۱	مربع‌های اعداد صحیح
۶۳	غیر قابل تغییر بودن و معماری
۶۴	تفکیک قابل تغییر بودن
۶۶	منشایی رویداد
۶۸	نتیجه‌گیری

بخش سوم: اصول طراحی

۷۳	فصل ۷: SRP اصل تک مسئولیتی
۷۴	نشانه ۱: تکرار تصادفی (اتفاقی)
۷۷	نشانه ۲: ادغام‌ها
۷۷	رادحل‌ها
۷۹	نتیجه‌گیری

۸۱	فصل ۸: OCP اصل
۸۲	یک آزمایش فکری
۸۶	کنترل جهت‌دار
۸۷	پنهان کردن اطلاعات

نتیجه گیری

۸۷

فصل ۹ LSP: اصل جایگزینی لیسکوف

۸۹

راهنمای استفاده از وراثت

۹۰

مساله مربع/مستطیل

۹۰

LSP و معماری

۹۱

مثالی از نقض LSP

۹۲

نتیجه گیری

۹۴

فصل ۱۰ ISP: اصل تفکیک اینترفیس

۹۵

ISP و زبان

۹۶

ISP و معماری

۹۷

نتیجه گیری

۹۸

فصل ۱۱ DIP: اصل معکوس سازی وابستگی

۹۹

انتزاع پایدار

۱۰۰

Factoryها

۱۰۱

کامپوننت CONCRETE

۱۰۲

نتیجه گیری

۱۰۳

بفش چهارم: اصول کامپوننت

۱۰۷

فصل ۱۲: کامپوننتها

۱۰۸

خلاصه‌ای از تاریخچه کامپوننتها

۱۱۱

وابستگی

۱۱۱

لینکرها

۱۱۳

نتیجه گیری

۱۱۵

فصل ۱۳: همبستگی کامپوننت

۱۱۶

استفاده مجدد / اصل هم‌ارزی استفاده مجدد

۱۱۷

اصل بستار مشترک

فهرست مطالب

شماره صفحه

عنوان

۱۱۸	اصل استفاده مجدد مشترک
۱۲۰	نمودار بحران برای همبستگی کامپوننت
۱۲۱	نتیجه گیری

فصل ۱۴: کامپوننت COUPLING

۱۲۳	اصل وابستگی غیر مدور
۱۲۳	طراحی بالا-پایین
۱۳۰	اصل وابستگی پایینی
۱۳۱	اصل انتزاع پایدار
۱۳۸	نتیجه گیری

افش پنجم: معماری

فصل ۱۵: معماری چیست؟

۱۴۹	توسعه
۱۵۰	استقرار
۱۵۱	عملکرد
۱۵۱	نگهداری
۱۵۲	نگهداری گزینه های آشکار
۱۵۳	مستقل شدن دستگاه
۱۵۵	نامۀ JUNK
۱۵۶	آدرس فیزیکی
۱۵۸	نتیجه گیری

فصل ۱۶: استقلال

۱۶۱	Use case ها
۱۶۲	عمل
۱۶۳	توسعه
۱۶۳	استقرار
۱۶۴	ترک گزینه های واضح

۱۶۴	لایه‌های جداسازی
۱۶۵	جداسازی USE CASE ها
۱۶۶	حالت جداسازی
۱۶۷	توانایی توسعه مستقل
۱۶۷	استقرار مستقل
۱۶۸	کپی
۱۶۹	جداسازی حالت ۱۰ (از نو)
۱۷۱	نتیجه گیری
۱۷۳	فصل ۱۷: خط مرزی، خطوط داخلی
۱۷۴	یک جفت از داستان‌های غمانگیر
۱۷۷	FITNESSE
۱۸۰	کدام خطوط را رسم می‌کنید و چه زمانی آن خطوط را رسم می‌کنید؟
۱۸۳	درباره ورود و خروج چطور؟
۱۸۴	معماری پلاگین
۱۸۶	استدلال پلاگین
۱۸۷	نتیجه گیری
۱۸۹	فصل ۱۸: تشریح خط مرزی
۱۸۹	تقاطع خط مرزی
۱۹۰	ترس یکنواخت
۱۹۲	استقرار کامپوننت‌ها
۱۹۳	Thread ما
۱۹۳	فرآیندهای محلی
۱۹۴	سرویس‌ها
۱۹۵	نتیجه گیری
۱۹۷	فصل ۱۹: قدامت‌ش و سطح
۱۹۸	سطح

فهرست مطالب

شماره صفحه

عنوان

۲۰۱	نتیجه گیری
۲۰۳	فصل ۱۰: قوانین کسب و کار
۲۰۴	موجودیت‌ها
۲۰۵	Use Case ها
۲۰۷	مدل‌های درخواست و پاسخ
۲۰۸	نتیجه گیری
۲۰۹	فصل ۱۱: معماری شکفت انگیز
۲۱۰	موضوع معماری
۲۱۰	هدف از یک معماری
۲۱۱	اما اهداف وب چیست؟
۲۱۱	فریم‌ورک‌ها ابزار هستند نه روش زندگی
۲۱۲	ساختار معماری‌های تست پذیر
۲۱۲	نتیجه گیری
۲۱۳	فصل ۱۲: معماری تمیز
۲۱۵	قانون وابستگی
۲۲۰	یک سناریوی متداول
۲۲۱	نتیجه گیری
۲۲۳	فصل ۱۳: Humble و شی Presenter
۲۲۴	الگوی شی Humble
۲۲۴	Presenter ها و View ها
۲۲۵	تست و معماری
۲۲۵	gateways های پایگاه داده
۲۲۶	DATA MAPPERS
۲۲۷	SERVICE LISTENERS
۲۲۷	نتیجه گیری

فهرست مطالب

عنوان

شماره صفحه

۲۲۹	فصل ۲۴: مرزهای جزئی
۲۳۰	پرش به مرحله آخر
۲۳۱	مرزهای یک بُعدی
۲۳۲	نماهای خارجی
۲۳۳	نتیجه گیری
۲۳۵	فصل ۲۵: لایه ها و مرزها
۲۳۶	شکار وامپوس
۲۳۷	معماری ؟
۲۴۰	عبور از جریان ها
۲۴۱	تقسیم جریان ها
۲۴۳	نتیجه گیری
۲۴۵	فصل ۲۶: کامپوننت اصلی
۲۴۶	جزئیات نهایی
۲۵۰	نتیجه گیری
۲۵۱	فصل ۲۷: سرویس ها، بزرگ و کوچک
۲۵۱	معماری سرویس؟
۲۵۲	مزایای سرویس؟
۲۵۴	مسئله بچه گریه
۲۵۶	اهداف مربوط به راهی (RESCUE)
۲۵۷	سرویس های مبتنی بر کامپوننت
۲۵۹	نگرانی های CROSS-CUTTING
۲۶۰	نتیجه گیری
۲۶۱	فصل ۲۸: مرز تست
۲۶۲	تست ها به عنوان کامپوننت های سیستم
۲۶۳	طراحی برای تست پذیری
۲۶۴	تست API

فهرست مطالب

شماره صفحه

عنوان

۲۶۵ نتیجه گیری

۲۶۷ فصل ۲۹: معماری توکار تمیز

۲۷۰ تست APP-TITUDE

۲۷۴ تنگراه سخت افزار - هدف

۲۸۶ نتیجه گیری

بخش ششم: جزئیات

۲۸۹ فصل ۳۰: بازه اطلاعات به عنوان جزئی از یک کل

۲۹۰ بانک اطلاعات رابطه‌ای

۲۹۰ چرا سیستم‌های بانک اطلاعات به این شکل هستند؟

۲۹۲ اگر دیسکی وجود نداشته باشد چه اتفاقی می‌افتد؟

۲۹۳ جزئیات

۲۹۳ اما در مورد عملکرد چه اتفاقی می‌افتد؟

۲۹۳ داستان کوتاه

۲۹۵ نتیجه گیری

۲۹۷ فصل ۳۱: وب یک جزء از کل است

۲۹۸ نوسان بی‌پایان

۳۰۰ نتیجه‌ی نهایی

۳۰۱ نتیجه گیری

۳۰۳ فصل ۳۲: فریم‌ورک‌ها اجزای معماری هستند

۳۰۴ عرضه کنندگان فریم‌ورک

۳۰۴ اتحاد نامتقارن

۳۰۵ ریسک‌ها

۳۰۶ راه حل

۳۰۷ هم اکنون به شما اعلام می‌کنم...

۳۰۷ نتیجه گیری

فهرست مطالب

شماره صفحه

عنوان

۳۰۹	فصل ۱۳۱۳: مطالعات موردی؛ فروش ویدئو
۳۱۰	محصول
۳۱۰	تحلیل use case
۳۱۲	معماری کامپوننت
۳۱۴	مدیریت وابستگی
۳۱۴	نتیجه گیری
۳۱۷	فصل ۱۳۱۴: مدل ۵M شده
۳۱۸	پکیج کردن به وسیله لایه بندی
۳۲۰	پکیج براساس ویژگی
۳۲۲	پورت ها و ADAPTERS
۳۲۵	پکیج براساس کامپوننت
۳۳۰	مشکل در جزئیات پیاده سازی است
۳۳۱	سازماندهی به جای کپسوله سازی
۳۳۵	دیگر روش های تفکیک
۳۳۷	نتیجه گیری: توصیه های به جا مانده
	بخش هفتم: پایانه
۳۴۱	پیوست A: تاریخچه معماری

وقتی در مورد معماری صحبت می‌کنیم، در مورد چه چیزی صحبت می‌کنیم؟

مانند هر توصیفی، توصیف نرم‌افزار از دیدگاه معماری به همان اندازه که گویا است، مبهم هم است. در معماری می‌توانید بیش از حد توان خودتان قول انجام کاری را بدهید که در نهایت قادر به تحویل آن کار باشید و در شرایط دیگر می‌توانید بیش از آنچه قول داده‌اید را تحویل دهید. زیبایی معماری در ساختار آن است و ساختار چیزی است که پارادایم‌ها^۱ و بحث‌های مربوط به توسعه نرم‌افزار مانند کامپوننت‌ها، کلاس‌ها، توابع، ماژول‌ها، لایه‌ها و خدمات، میکرو یا ماکرو را به خود اختصاص داده است. اما ساختار بدشکل بسیاری از سیستم‌های نرم‌افزاری معمولاً برعکس باور یا درک ما هستند، مانند طرح‌های سازمانی به سبب «تصادیه جماعی شوروی» بلوک‌های چوبی بازی جنگا^۲ که رسیدنشان به ابر غیر محتمل است، بخش‌های قشری از یک سیستم نرم‌افزاری که معماری قابل فهم ندارد^۳ و فراموش شده‌اند. واضح نیست که ساختار نرم‌افزار به همان روش ساختار ساختمان، از برداشت ذهنی ما پیروی می‌کند.

ساختمان‌ها دارای یک ساختار فیزیکی هستند، خواه ساختمان پایه و اساس آن‌ها در سنگ باشد یا در بتن، خواه ساختمان‌ها دارای یک طاق بلند باشند (چند طبقه و بلند) یا فضای سرتاسری و جادار داشته باشند (یک طبقه و دارای فضای بزرگ و وسیع)، خواه ساختمان‌ها بزرگ باشند یا کوچک، چه مجلل باشند یا بی‌زرق و برق. ساختار ساختمان‌ها انتخاب‌های کم و محدودی هستند، اما همه این ساختارها مطابق با قوانین فیزیکی گرانش و قوانین فیزیکی موادشان هستند. از سوی دیگر، در مورد نرم‌افزار، زمان کمی برای مطابقت با قوانین طراحی اصلی، صرف می‌شود البته به جز در مواردی که نرم‌افزار دارای اهمیت باشد. نرم‌افزار از چه چیزی ساخته شده است؟ برخلاف ساختمان‌هایی که ممکن است از آجر، بتن، چوب، فولاد و شیشه ساخته شوند، نرم‌افزار

1. Paradigms

۲. بازی جنگا Jenga از ۵۴ بلوک چوبی ساخته شده است که طول هر بلوک باید دقیقاً سه برابر عرض و ضخامتش هم باید یک پنجم طولش باشد. برای شروع بازی جنگا Jenga باید بلوک‌های چوبی رو به صورت افقی و عمودی روی هم بچینید تا یک برج ۱۸ طبقه چوبی ساخته بشود. بعد از اینکه برج چوبی تکمیل شد، بازیکنی که برج را چیده باید بازی را شروع کند. روش بازی به این صورت است که هر بازیکن به نوبت باید یک بلوک چوبی را از هر جایی غیر از دو ردیف اول (بالای برج) بردارد. هر بازیکنی که برج را خراب کند، باخته است! «مترجم»

۳. big ball of mud یک سیستم نرم‌افزاری است که معماری قابل فهم ندارد. اگر چه از دیدگاه مهندسی نرم‌افزار ناخوشایند است، اما چنین سیستم‌هایی به دلیل فشارهای تجاری، بازده توسعه‌دهنده و آنتروپی که در عمل کاربرد دارند. آن‌ها یک نوع ضد الکوی طراحی هستند. «مترجم»

از نرم‌افزار ساخته شده است. ساختارهای نرم‌افزاری بزرگ از کامپوننت‌های نرم‌افزاری کوچک‌تر ساخته می‌شوند که به نوبه خود همچنان از کامپوننت‌های نرم‌افزاری کوچک‌تری ساخته شده‌اند و همین‌طور ادامه می‌یابد. کدنویسی می‌تواند شبیه یک سلسله مراتب وابسته به یکدیگر باشد.

زمانی که در مورد معماری نرم‌افزار صحبت می‌کنیم، نرم‌افزار ذاتاً بازگشتی و فراکتال است، طرح نرم‌افزار در کد ریخته می‌شود. همه چیز مربوط به جزئیات است. سطوح در هم تنیده جزئیات، در معماری ساختمان نیز وجود دارد، اما صحبت کردن در مورد واحد سنجش فیزیکی، در نرم‌افزار بی‌معنی است. نرم‌افزار دارای ساختارهای بسیاری است - حتی انواع بسیاری از ساختارها را دارد - با این حال تنوع ساختاری نرم‌افزاری، بیشتر از ساختارهای فیزیکی موجود در ساختمان‌ها، می‌باشد. شما حتی می‌توانید به درست‌تر بگویید، کنید که در معماری نرم‌افزار نسبت به معماری ساختمان، فرآیندهای طراحی و تمرکز بیشتر وجه دارد، بدین معنا که غیرمنطقی نیست وابستگی به معماری در معماری نرم‌افزار، بیشتر از وابستگی به معماری در معماری ساختمان باشد.

اما مقیاس فیزیکی چیزی است که آن‌ها در جهان می‌فهمند و به دنبال آن هستند.

اگر چه از لحاظ دیداری جذاب است، اما سطوحی که موجود در یک نمودار پاورپوینت، معماری یک سیستم نرم‌افزاری نیستند. شکی نیست که آن‌ها معماری را از یک دیدگاه خاص نمایش می‌دهند، اما اشتباه گرفتن این مستطیل‌ها با تصویر کلی برای معماری از دست دادن تصویر کلی در معماری است (اگر دچار چنین اشتباهی شویم، مفهوم معماری را به همانی دیگر نکردیم). معماری نرم‌افزار شبیه هیچ چیز نیست. نشان دادن با شکل‌ها، تنها یک انتخاب است، یک روش مشخص و معلوم. این کار یک انتخاب براساس یک مجموعه از انتخاب‌هاست: (انتخاب‌هایی که باید شامل چه چیزی باشد، باید چه چیزی را حذف کرد، با استفاده از شکل یا رنگ روی چه چیزی باید تأکید کرد، از طریق هم شکلی یا نادیده گرفتن روی چه چیزی نباید تأکید کرد. هیچ چیز ذاتی یا طبیعتاً مورد برتری یک دیدگاه نسبت به دیگری وجود ندارد.

هر چند که ممکن است صحبت کردن در مورد فیزیک و مقیاس فیزیکی در معماری نرم‌افزار بی‌معنی باشد، ما نگران برخی محدودیت‌های فیزیکی خاص هستیم. سرعت پردازنده و پهنای باند شبکه می‌توانند تاثیر شدیدی بر بازدهی و عملکرد نرم‌افزار داشته باشند. حافظه و رسانه ذخیره‌سازی می‌توانند

زیاده‌خواهی‌های هر code base را محدود کنند. نرم‌افزار می‌تواند ابزاری باشد که ما با استفاده از آن رویاهایمان را عملی می‌کنیم، اما در نهایت در جهان فیزیکی اجرا می‌شود و محدود است. این شرارت و بی‌عاطفگی در عشق است، بانو، که آرزو بی‌نهایت است و اجرا محدود است؛ میل بی‌حد و حصر است و عمل یک برده محدود است.

– قسمتی از نمایش نامه تراژیک ترویلوس و کرسیدا (انگلیسی: Troilus and Cressida) اثر ویلیام شکسپیر Act III, Scene II.

جهان فیزیکی جایب است که ما و شرکت‌هایمان و اقتصادهایمان در آن زندگی می‌کنند. این یک درجه‌بندی دیگر با ما می‌باشد که ما می‌توانیم معماری نرم‌افزار را با استفاده از آن درک کنیم، به جای دیگر نیروها و کمیت‌های فیزیکی که ما می‌توانیم از طریق آن‌ها صحبت و استدلال کنیم. معماری، تصمیمات مهم انسانی را می‌دهد که یک سیستم را شکل می‌دهد، در این جا مهم بودن، با هزینه تغییر، اندازه‌گیری می‌شود.

– گریدی بوش

زمان، پول و تلاش (نیروی انسانی)، درکی از مقیاس ما می‌دهند که برای تمایز قابل شدن بین بزرگ بودن و کوچک بودن نرم‌افزار و امکان تشخیص ساخت معماری از چیزهای دیگر را برای ما فراهم می‌کنند. این اندازه‌گیری همچنین به ما می‌گوید که چقدر می‌توانیم مشخص کنیم که آیا معماری خوب است یا خیر. یک معماری خوب نه تنها نیازهای کاربران، توسعه‌دهندگان و مالکان را در یک نقطه معین از زمان برآورده می‌کند، بلکه با گذشت زمان نیز نیازهای آن‌ها را برآورده می‌کند. اگر فکر می‌کنید معماری خوب گران است، معماری بد را امتحان کنید.

۱. در توسعه نرم‌افزار، codebase یا (code base) به کل مجموعه‌ای از کد منبع اشاره می‌کند که برای ساخت یک سیستم نرم‌افزاری، نرم‌افزار یا component نرم‌افزاری خاص استفاده می‌شود. به طور معمول، codebase شامل تنها فایل‌های کد منبعی نوشته شده توسط انسان است. بنابراین، codebase معمولاً شامل کد منبع ایجاد شده از طریق ابزار (فایل تولید شده) یا کتابخانه‌های دودویی (object files) نیست، زیرا آن‌ها می‌توانند از source code اولیه نوشته شده توسط انسان ساخته شوند. با این حال، به طور کلی شامل پوشه‌های configuration و property files می‌شود، زیرا آن‌ها اطلاعات لازم برای ساخت هستند. «مترجم»