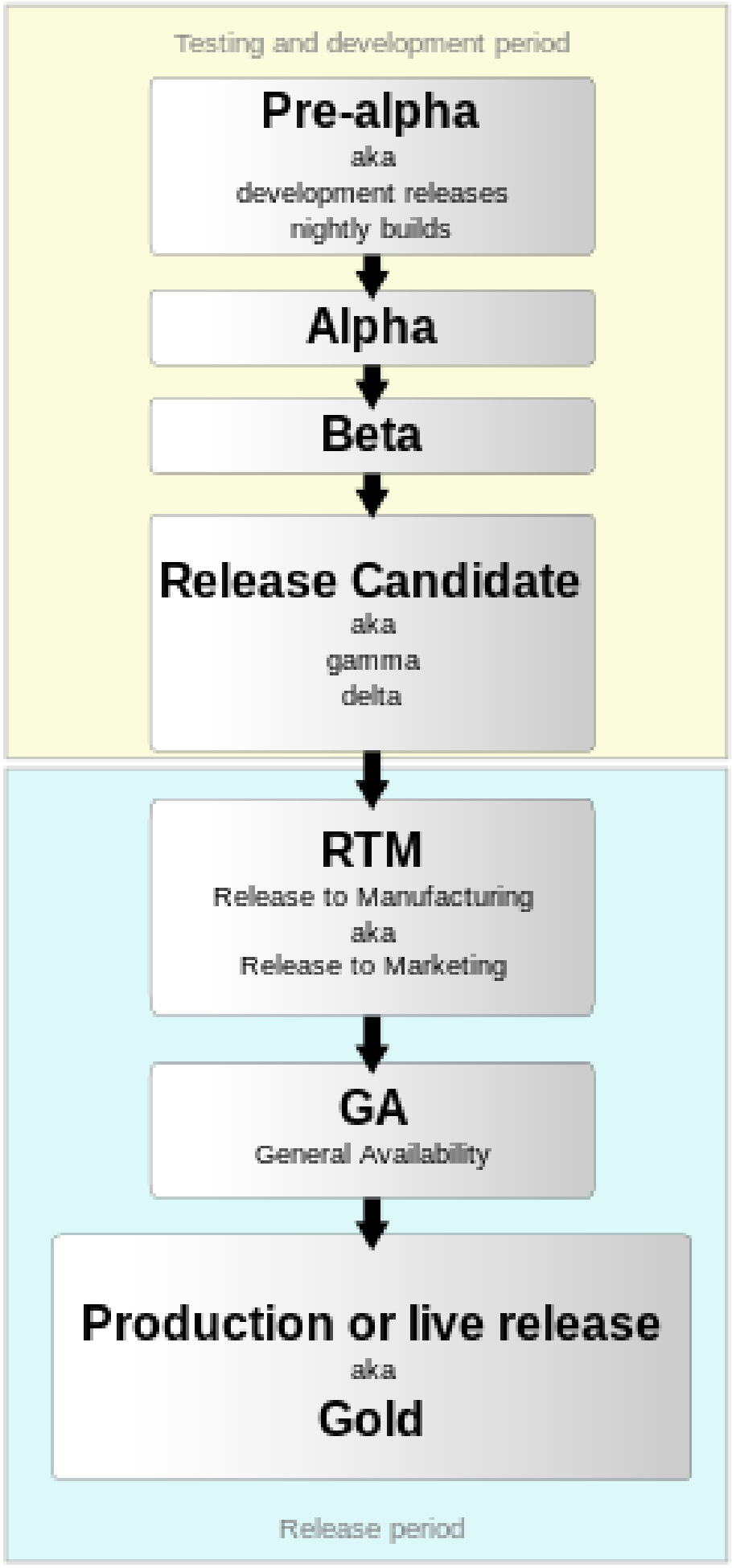
عنوان: **آشنایی با نسخه بندی و چرخه انتشار نرم افزارها**
نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۲/۰۵/۰۵ ۲۳:۰
آدرس: www.dotnettips.info
برچسب‌ها: versioning, Software Development, version control, Assembly

نسخه بندی و چرخه انتشار یک نرم افزار، اهمیت زیادی در ارائه یک نرم افزار خوب دارد. هر چه نرم افزار شما بزرگ‌تر و از کتابخانه‌های بیشتری در تولید آن استفاده شده باشد، در بروز رسانی و نسخه بندی آن دقت بیشتری باید داشت و کار دشوارتری است. اما چگونه به بهترین روش، نسخه بندی نرم افرار خود را مدیریت نمایید.

مقدمه:

حتما نسخه بندی و نگارش‌های مختلف نرم افزارهایی را که استفاده می‌کنید، مشاهده نموده‌اید. نسخه‌های آلفا یا بتا یا نسخه بندی سالیانه یا با حروف و اعداد خاص. با این حال همه نرم افزارها علاوه بر عناوین متعارف، یک نسخه بندی داخلی عددی، شماره‌ای هم دارند. بسته به حجم و اندازه نرم افزارها، ممکن چرخه انتشار نرم افزارها متفاوت باشند. سیاست عرضه نرم افزار در هر شرکت هم متفاوت است. مثلا شرکت مایکروسافت برای عرضه ویندوز ابتدا نسخه بتا یا پیش نمایش آن را عرضه نموده تا با دریافت بازخوردهایی از استفاده کنندگان، نسخه نهایی نرم افزار خود را با حداقل ایراد و خطا عرضه نماید. البته این بخاطر بزرگی نرم افزار ویندوز نیز می‌باشد اما شرکت ادوبی اکثرا هر یکی دو سال بدون عرضه نسخه‌های قبل از نهایی یک دفعه نسخه جدیدی را رسما عرضه می‌نماید.


Testing and development period
Pre-alpha
aka
development releases
nightly builds
Alpha
Beta
Release Candidate
aka
gamma
delta
RTM
Release to Manufacturing
aka
Release to Marketing
GA
General Availability
Production or live release
aka
Gold
Release period

چرخه انتشار نرم افزار:

چرخه انتشار نرم افزار از زمان شروع کد نویسی تا عرضه نسخه نهایی می‌باشد که شامل چندین مرحله و عرضه نرم افزار می‌باشد.

Pre-alpha

این مرحله شامل تمام فعالیت‌های انجام شده قبل از مرحله تست می‌باشد. در این دوره آنالیز نیازمندیها، طراحی نرم افزار، توسعه نرم افزار و حتی تست واحد باشد. در نرم افزارهای سورس باز چندین نسخه قبل از آلفا ممکن است عرضه شوند.

Alpha

این مرحله شامل همه فعالیت‌ها از زمان شروع تست می‌باشد. البته منظور از تست، تست تیمی و تست خود نرم افزار می‌باشد. نرم افزارهای آلفا هنوز ممکن است خطا و اشکالاتی داشته باشند و ممکن است اطلاعات شما از بین رود. در این مرحله امکانات جدیدی مرتبا به نرم افزار اضافه می‌گردد.

Beta

نرم افزار بتا، همه قابلیت‌های آن تکمیل شده و خطاهای زیادی برای کامل شدن نرم افزار وجود دارد. در این مرحله بیشتر به تست کاهش تاثیرات به کاربران و تست کارایی دقت می‌شود. نسخه بتا، اولین نسخه‌ای خواهد بود که بیرون شرکت و یا سازمان در دسترس قرار می‌گیرد. برخی توسعه دهندگان به این مرحله preview، technical preview یا early access نیز می‌گویند.

Release candidate

در این مرحله نرم افزار، آماده عرضه به مصرف کنندگان است و نرم افزارهایی مثل سیستم عامل‌های ویندوز در دسترس تولید کنندگان قرار گرفته تا با جدیدترین سخت افزار خود یکپارچه شوند.

(General availability (GA

در این مرحله، عرضه عمومی نرم افزار و بازاریابی و فروش نرم افزار مد نظر است و علاوه بر این تست امنیتی و در نرم افزارهای خیلی بزرگ عرضه جهانی صورت می‌گیرد

مراحلی همچون عرضه در وب و پشتیبانی نیز وجود دارند.

نسخه بندی نرم افزار:

برنامه‌های ویندوزی یا وب در ویژوال استادیو یک فایل AssemblyInfo دارند که در قسمت آخر آن، اطلاعات مربوط به نسخه نرم افزار ذخیره می‌شود. هر نسخه نرم افزار شامل چهار عدد می‌باشد که با نقطه از هم جدا شده است.

Major Version

وقتی افزایش می‌یابد که تغییرات قابل توجهی در نرم افزار ایجاد شود

Minor Version

وقتی افزایش یابد که ویژگی جزئی یا اصلاحات قابل توجهی به نرم افزار ایجاد شود.

Build Number

به ازای هر بار ساخته شدن پروژه افزایش می‌یابد.

Revision

وقتی افزایش می‌یابد که نواقص و باگ‌های کوچکی رفع شوند.

وقتی که major یا minor افزایش یابد می‌تواند با کلماتی همچون alpha، beta یا release candidate همراه شود.در اکثر برنامه‌های تجاری اولین شمارهٔ انتشار یک محصول از نسخهٔ شمارهٔ یک شروع می‌شود. ترتیب نسخه بندی هم ممکن است تغییر یابد

```
major.minor[.build[.reversion]]
```

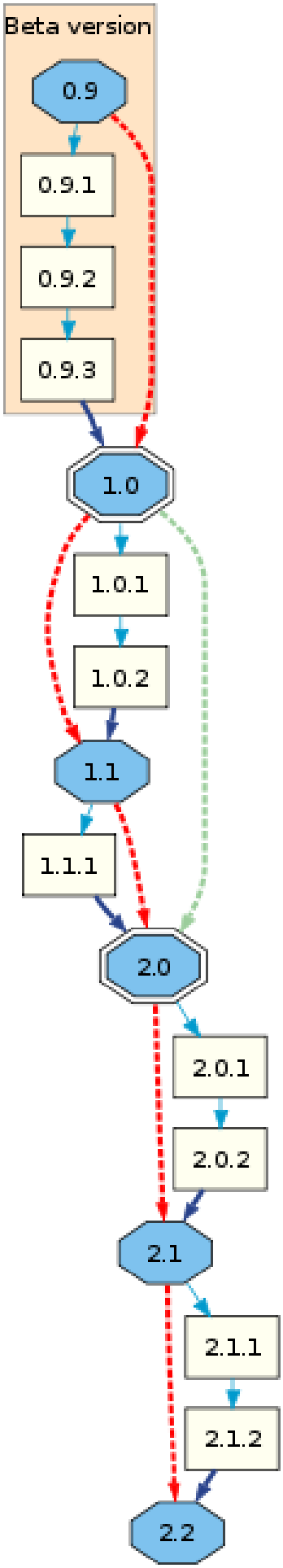
یا

```
major.minor[.maintenance[.build]]
```

نسخه بندی مایکروسافت:

اگر به نسخه برنامه Office توجه کرده باشید مثلا Office 2013 نسخه 15.0.4481.1508 می‌باشد که در این روش از تاریخ شروع پروژه و تعداد ماه‌ها یا روزها و یا ثانیه‌ها با یک الگوریتم خاص برای تولید نسخه نرم افزار استفاده می‌شود.

نسخه بندی معنایی:


Beta version
0.9
0.9.1
0.9.2
0.9.3
1.0
1.0.1
1.0.2
1.1
1.1.1
2.0
2.0.1
2.0.2
2.1
2.1.1
2.1.2
2.2

به عنوان یک راه حل، مجموعه‌ی ساده‌ای از قوانین و الزامات که چگونگی طراحی شماره‌های نسخه و افزایش آن را مشخص می‌کند، وجود دارد. برای کار کردن با این سیستم، شما ابتدا نیاز به اعلام API عمومی دارید. این خود ممکن است شامل مستندات و یا اجرای کد باشد.

علیرغم آن، مهم است که این API، روشن و دقیق باشد. هنگامیکه API عمومی خود را تعیین کردید، تغییرات برنامه شما بر روی نسخه API عمومی تاثیر خواهد داشت و آنرا افزایش خواهد داد. بر این اساس، این مدل نسخه‌بندی را در نظر بگیرید: X.Y.Z یعنی (Major.Minor.Patch).

رفع حفره‌هایی که بر روی API عمومی تاثیر نمی‌گذارند، مقدار Patch را افزایش می‌دهند، تغییرات جدیدی که سازگار با نسخه قبلی است، مقدار Minor را افزایش می‌دهند و تغییرات جدیدی که کاملا بدیع هستند و به نحوی با تغییرات قبلی سازگار نیستند مقدار Major را افزایش می‌دهند.

نرم‌افزارهایی که از نسخه بندی معنایی استفاده می‌کنند، باید یک API عمومی داشته باشند. این API می‌تواند در خود کد یا و یا به طور صریح در مستندات باشد که باید دقیق و جامع باشد.
یک شماره نسخه صحیح باید به شکل X.Y.Z باشد که در آن X،Y و Z اعداد صحیح غیر منفی هستند. X نسخه‌ی Major می‌باشد، Y نسخه‌ی Minor و Z نسخه‌ی Patch می‌باشد. هر عنصر باید یک به یک و بصورت عددی افزایش پیدا کند. به عنوان مثال: 1.9.0 -> 1.10.0 <- 1.11.0
هنگامی که به یک نسخه‌ی Major یک واحد اضافه می‌شود، نسخه‌ی Minor و Patch باید به حالت 0 (صفر) تنظیم مجدد گردد.
هنگامی که به شماره نسخه‌ی Minor یک واحد اضافه می‌شود، نسخه‌ی Patch باید به حالت 0 (صفر) تنظیم مجدد شود. به عنوان مثال: 1.1.3 -> 2.0.0 و 2.1.7 -> 2.2.0
هنگامیکه یک نسخه از یک کتابخانه منتشر می‌شود، محتوای کتابخانه مورد نظر نباید به هیچ وجه تغییری داشته باشد. هر گونه تغییر جدیدی باید در قالب یک نسخه جدید انتشار پیدا کند.
نسخه‌ی Major صفر (Y.Z.0) برای توسعه‌ی اولیه است. هر چیزی ممکن است در هر زمان تغییر یابد. API عمومی را نباید پایدار در نظر گرفت.
نسخه 1.0.0 در حقیقت API عمومی را تعریف می‌کند. چگونگی تغییر و افزایش هر یک از نسخه‌ها بعد از انتشار این نسخه، وابسته به API عمومی و تغییرات آن می‌باشد.
نسخه Patch یا (x.y.Z | x > 0) فقط در صورتی باید افزایش پیدا کند که تغییرات ایجاد شده در حد برطرف کردن حفره‌های نرم‌افزار باشد. برطرف کردن حفره‌های نرم‌افزار شامل اصلاح رفتارهای اشتباه در نرم‌افزار می‌باشد.
نسخه Minor یا (x.Y.z | x > 0) فقط در صورتی افزایش پیدا خواهد کرد که تغییرات جدید و سازگار با نسخه قبلی ایجاد شود. همچنین این نسخه باید افزایش پیدا کند اگر بخشی از فعالیت‌ها و یا رفتارهای قبلی نرم‌افزار به عنوان فعالیت منقرض شده اعلام شود. همچنین این نسخه می‌تواند افزایش پیدا کند اگر تغییرات مهم و حیاتی از طریق کد خصوصی ایجاد و اعمال گردد. تغییرات این نسخه می‌تواند شامل تغییرات نسخه Patch هم باشد. توجه به این نکته ضروری است که در صورت افزایش نسخه Minor، نسخه Patch باید به 0 (صفر) تغییر پیدا کند.
نسخه Major یا (X.y.z | X > 0) در صورتی افزایش پیدا خواهد کرد که تغییرات جدید و ناهمخوان با نسخه فعلی در نرم‌افزار اعمال شود. تغییرات در این نسخه می‌تواند شامل تغییراتی در سطح نسخه Minor و Patch نیز باشد. باید به این نکته توجه شود که در صورت افزایش نسخه Major، نسخه‌های Minor و Patch باید به 0 (صفر) تغییر پیدا کنند.
یک نسخه قبل از انتشار می‌تواند توسط یک خط تیره (dash)، بعد از نسخه Patch (یعنی در انتهای نسخه) که انواع با نقطه (dot) از هم جدا می‌شوند، نشان داده شود. نشان‌گر نسخه قبل از انتشار باید شامل حروف، اعداد و خط تیره باشد [0-9A-Za-z-]. باید به این نکته دقت داشت که نسخه‌های قبل از انتشار خود به تنهایی یک انتشار به حساب می‌آیند اما اولویت و اهمیت نسخه‌های عادی را ندارد. برای مثال: 1.0.0-x.7.z.92 ، 1.0.0-0.3.7 ، 1.0.0-alpha.1 ، alpha
یک نسخه Build می‌تواند توسط یک علامت مثبت (+)، بعد از نسخه Patch یا نسخه قبل از انتشار (یعنی در انتهای نسخه) که انواع آن با نقطه (dot) از هم جدا می‌شوند، نشان داده شود. نشان‌گر نسخه Build باید شامل حروف، اعداد و خط تیره باشد [0-

-9A-Za-z]. باید به این نکته دقت داشت که نسخه‌های Build خود به تنهایی یک انتشار به حساب می‌آیند و اولویت و اهمیت بیشتری نسبت به نسخه‌های عادی دارند. برای مثال: 1.0.0+build.11.e0f985a ، 1.3.7+build.1
اولویت‌بندی نسخه‌ها باید توسط جداسازی بخش‌های مختلف یک نسخه به اجزای تشکیل دهنده آن یعنی Minor، Major، Patch، نسخه قبل از انتشار و نسخه Build و ترتیب اولویت بندی آن‌ها صورت گیرد. نسخه‌های Minor، Major و Patch باید بصورت عددی مقایسه شوند. مقایسه نسخه‌های قبل از انتشار و نسخه Build باید توسط بخش‌های مختلف که توسط جداکننده‌ها (نقطه‌های جداکننده) تفکیک شده است، به این شکل سنجیده شود:

بخش‌هایی که فقط حاوی عدد هستند، بصورت عددی مقایسه می‌شوند و بخش‌هایی که حاری حروف و یا خط تیره هستند بصورت الفبایی مقایسه خواهند شد.

بخش‌های عددی همواره اولویت پایین‌تری نسبت به بخش‌های غیر عددی دارند. برای مثال:

1.0.0-alpha < 1.0.0-alpha.1 < 1.0.0-beta.2 < 1.0.0-beta.11 < 1.0.0-rc.1 < 1.0.0-rc.1+build.1 < 1.0.0 < 1.0.0+0.3.7 < 1.3.7+build < 1.3.7+build.2.b8f12d7 < 1.3.7+build.11.e0f985a

منبع نسخه بندی معنایی : semver.org

نظرات خوانندگان

نویسنده: میثم هوشمند
تاریخ: ۱۳۹۲/۰۷/۲۸ ۱۰:۲۱

در خصوص RTM توضیح خاصی ارائه نشده!

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۷/۲۹ ۱۸:۵۱

rtm هست در تصویر اول. به معنای release to manufacturing است. مثلا مایکروسافت اول ویندوز 8 رو در اختیار لپ تاپ سازها قرار می‌ده تا نصب کنند. بعد همون نگارش چند وقت بعد برای عموم توزیع میشه. اگر این RTM برای برنامه نویس‌ها باشه، یعنی نگارش نهایی که مثلا به دارندگان اکانت‌های MSDN اول ارائه شده. بعد از چند وقت همون توزیع‌ها با سریال آزمایشی در اختیار عموم قرار می‌گیرند. rtm به معنای کیفیتی از کار است که قابل ارائه است به عموم در سطح وسیع.

نویسنده: ناظم
تاریخ: ۱۳۹۲/۰۸/۲۵ ۱۳:۵۴

سلام
بنا بر این میتوان نتیجه گرفت که نسخه rtm همیشه همان نسخه ای هست که انتشار نهایی و عمومی میشه؟

عنوان: **به‌روز رسانی اسمبلی‌های خارجی**
نویسنده: میثم نوایی
تاریخ: ۱۳۹۳/۰۷/۲۷ ۱۶:۲۵
آدرس: www.dotnettips.info
گروه‌ها: Assembly, Project Management, batch file

در اکثر شرکت‌های بزرگ و متوسط نرم افزاری، بخش مشترکی از پروژه‌ها تحت عنوان فریم ورک و یا پروژه‌های مشترک (Common) از پروژه‌های جاری فاکتور گرفته میشود و ارتباط با آنها با ارجاعی (Reference) به اسمبلی آنها انجام مشود.
اما مشکل همیشگی این است که برای حفظ استقلال، مستقیما از پروژه‌های جاری به اسمبلی‌های پایه ارجاع داده نمی‌شود؛ چون ممکن است بنا بر پایسته بودن نسخه پروژه جاری، قصد نداشته باشیم همیشه آخرین ورژن اسمبلی‌های خارجی را دریافت کنیم، بلکه ارجاعی به اسمبلی‌ها در یک پوشه در خود پروژه انجام میشود و شما بعد از هر بار تغییر در فریم ورک، باید اسمبلی‌های جدید را به داخل پوشه، در تک تک پروژه‌های جاری تان کپی کنید.
برای خلاصی از این کار مدام و تکراری می‌توانید از یک Batch فایل شبیه کد زیر استفاده کنید:

```
xcopy /s /y D:\Project\Framework\Framework.Web\bin\Debug\Framework.dll D:\Project\Current\DependentDLL
xcopy /s /y D:\Project\Framework\Framework.Web\bin\Debug\Framework.Common.dll
D:\Project\Current\DependentDLL
xcopy /s /y D:\Project\Framework\Framework.Web\bin\Debug\Framework.Business.dll
D:\Project\Current\DependentDLL
xcopy /s /y D:\Project\Framework\Framework.Web\bin\Debug\Framework.Web.dll
D:\Project\Current\DependentDLL
```

کافی است این دستورات را در Note Pad کپی کنید و سپس با پسوند bat و مثلا با نام Update ذخیره کنید. این فایل را در پوشه اسمبلی‌های وابسته در پروژه‌های جاری تان کپی کنید و از این به بعد هر وقت خواستید آخرین ورژن اسمبلی‌های خارجی را دریافت کنید دوبار روی این فایل کلیک کنید. برای شخصی سازی بیشتر دستورات انتقال فایل در Batch فایل‌ها اینجا و اینجا را بخوانید.

نظرات خوانندگان

نویسنده: محسن موسوی
تاریخ: ۱۳۹۳/۰۷/۲۸ ۱۰:۲۱

البته استفاده از Nuget به صورت محلی گزینه‌ی بهتری هست. نوگت راه حل‌های مختلفی برای این کار ارئه میده. بصورت یک پوشه اشتراکی و یا ایجاد یک سرویس نوگت.

نویسنده: میثم نوایی
تاریخ: ۱۳۹۳/۰۷/۲۸ ۱۴:۱۷

بله استفاده از روش مطروحه شما استاندارد و اصولی‌تر میباشد.
راهکار من سر راست‌تر و سریع‌تر میباشد و برای افراد مبتدی کاربرد بیشتری دارد به اضافه اینکه خیلی از اوقات اسمبلی‌های خارجی در پروژه مدام دستخوش تغییر مسیر،تغییر نام و یا حتی ممکن است بکلی از پروژه کنار گذاشته شوند.