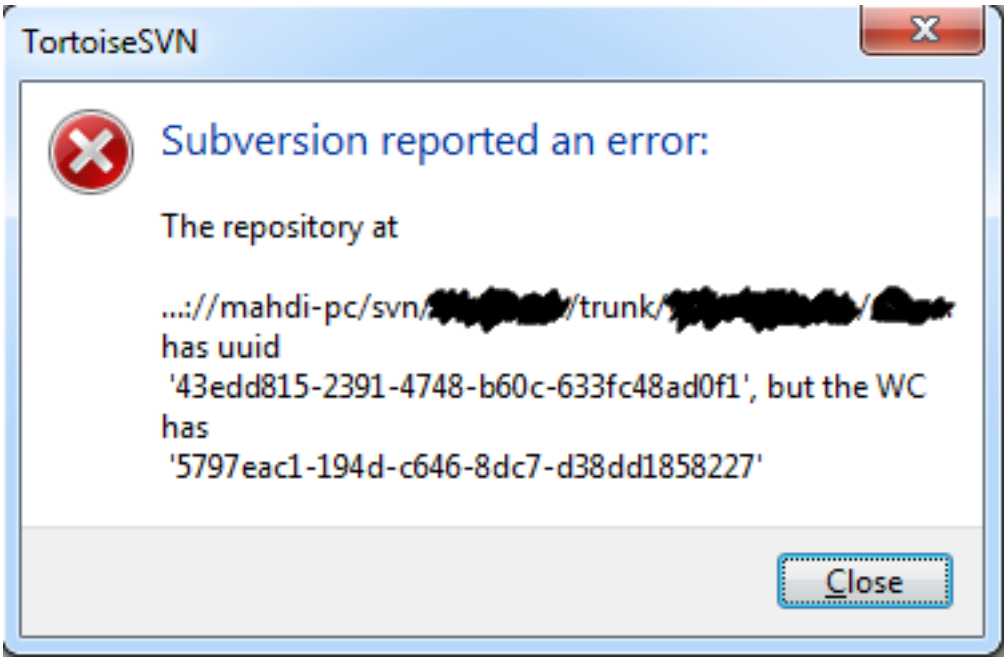
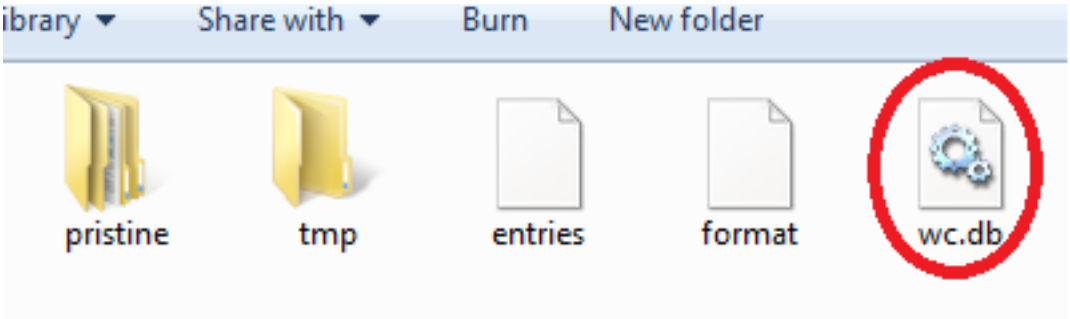
عنوان: **مشکل در جابجایی پروژه های svn**
نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۷/۱۹ ۱۴:۴۶
آدرس: www.dotnettips.info
برچسب‌ها: SVN, Subversion

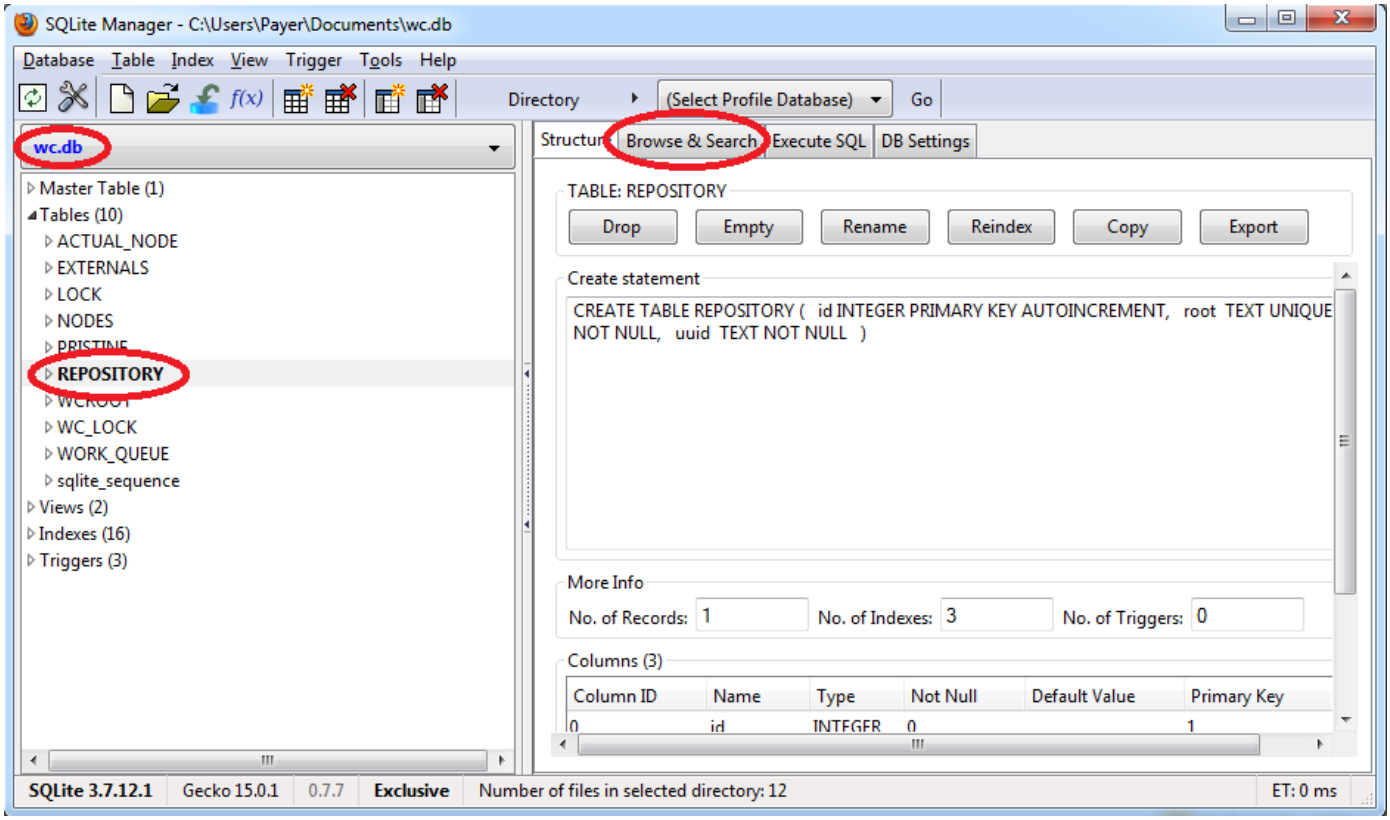
بنا به دلایلی، کدهای نگهداری شده توسط svn در سیستم خودم رو باید به سیستم دیگری وارد می‌کردم که شامل چند پروژه می‌شد و در قسمت مربوطه چند پروژه کار شده بود. تا این که مشکلی پیش آمد و جهت ادامه توسعه مجبور به برگرداندن پروژه‌ها به سیستم قبلی شدیم. حالا مشکل این بود که به repository قبلی هیچ گونه دسترسی نداشتیم و نمیخواستیم که کدهای در دست هم بصورت یک پروژه تازه وارد سرور svn شود. برای همین مجبور بودیم که از طریق relocate کردن، این کار را انجام دهیم، که به خطای زیر برخورد کردم:



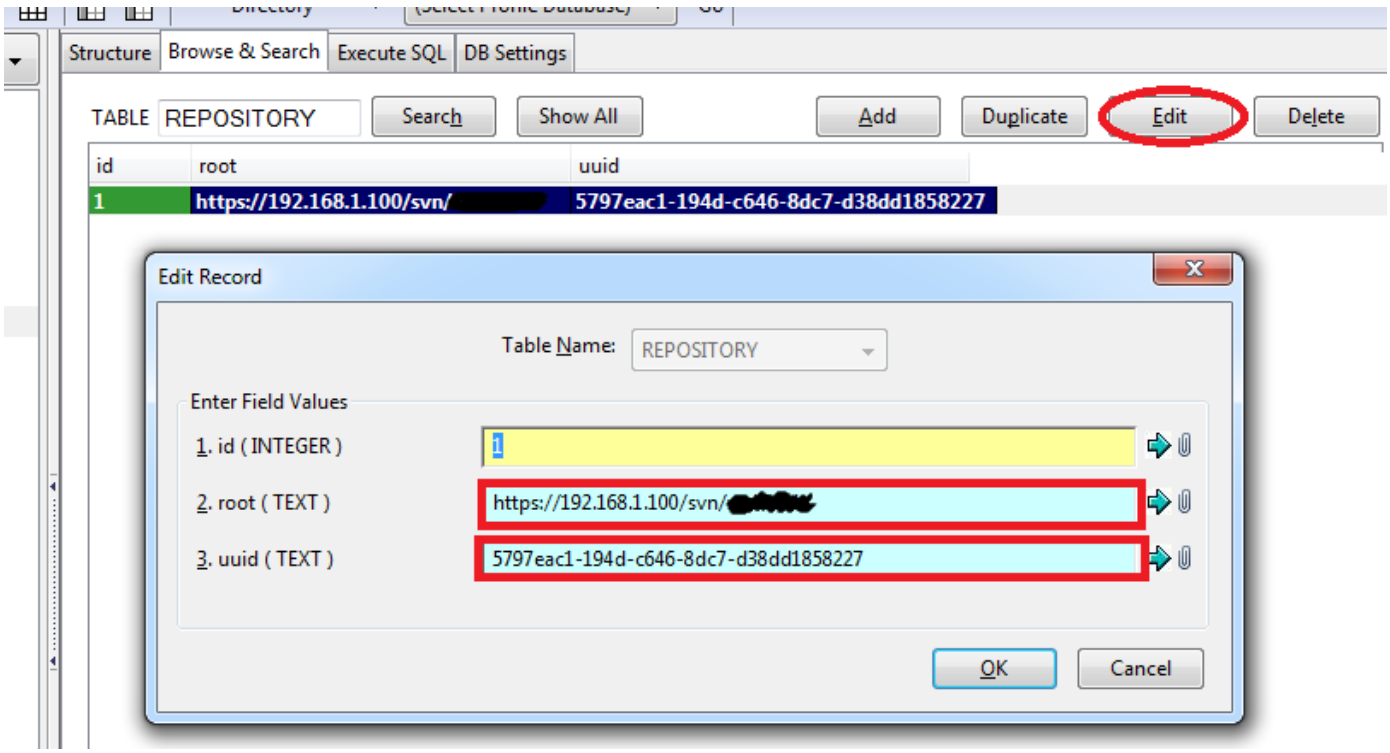
بله؛ مشکل از اینجا ناشی می‌شد که سرور دوم کدها را import نکرده بود و بلکه با یک کامیت ساده، کدها وارد سرور شده بودند.
راه حل: در کل برای تغییر این uid که در تصویر مشخص شده، باید فایل SQLite مربوطه را یافت و به راحتی، دو مشخصه مربوط به آدرس مخزن و uid را در آن اصلاح کرد.
در پوشه svn کنار پروژه، فایلی با نام wc.db از نوع SQLite وجود دارد:



که به سادگی می‌توان آن‌را با برنامه‌ای مانند افزونه ای در فایرفاکس گشود و ویرایش کرد:



همانطور که مشخص است، در جدول repository، تک ردیفی وجود دارد که مشخصات مربوط به مخزن svn در آن ذخیره شده است:



نکته: در این نوع آدرس دهی، باید حتما به نوع قرار گیری پروژه‌ها در کنار هم نیز دقت داشت. مثلا به محل آخری که در آدرس وجود دارد باید دقت کرد تا در مسیردهی، اشتباهی صورت نگیرد و فایلها جابجا کامیت نشوند.

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۲۳ ۱۳:۰

این مطلب امروز به درد من خورد!
پروژه iTextSharp محل مخزنش رو عوض کرده یا شاید هم Source forge دست به تغییر و تحول زده. محل جدید برای checkout شده:

```
svn checkout http://svn.code.sf.net/p/itextsharp/code/trunk itextsharp-code
```

و با uuid جدید زیر

```
820d3149-562b-4f88-9aa4-a8e61a3485cf
```

برای relocate مخزن قبلی به محل جدید حتما نیاز است این uuid را در بانک اطلاعاتی فوق ویرایش کرد.

عنوان: **ابزاری برای تغییر خودکار ارجاعات پروژه برای کار با SVN SourceControl**
نویسنده: بهزاد
تاریخ: ۱۳۹۲/۰۲/۰۲ ۱۱:۳۵
آدرس: www.dotnettips.info
برچسب‌ها: Subversion, SVN, Visual Studio, Software deployment, VisualSVN

ما در شرکت برای Source Control از SVN استفاده می‌کنیم، مزایای سورس کنترل آنقدر واضح است که دیگه من اینجا چیزی ازش نمیگم
اما برای استفاده از سورس کنترل یک مشکلی وجود دارد، اگر شما تعدادی پروژه را به کاربران خاصی بدین و تعدادی رو ندین، اون کاربر وقتی پروژه‌ها را می‌گیره با مشکل ارجاعات پروژه‌ها مواجه است. چرا که برخی از پروژه‌های ارجاعی، روی کامپیوتر برنامه نویس 1 وجود نداره. برعکس ش هم همین طوره، چون اون کاربر، پروژه‌های ارجاعی رو نداره، باید به جاش به اسمبلی نهایی اون پروژه ارجاع بده. بنابراین وقتی مدیر پروژه‌ها رو می‌گیره، باز ارجاعات اشتباه هستند!
ما اینجا برای رفع این مشکل ابزاری درست کردیم، به اسم SolutionExplorer.
این ابزار فایل solution رو به همراه پوشه حاوی فایل‌های اسمبلی می‌گیره. اگر پروژه ای به اسمبلی ای ارجاع داده باشه که پروژه اش توی solution باشه، ارجاع به اسمبلی رو تبدیل میکنه به ارجاع به پروژه و برعکسش، اگر پروژه ای به پروژه دیگه ای ارجاع داده باشه که توی solution وجود نداشته باشه، توی پوشه اسمبلی ها، دنبال اسمبلی ای می‌گرده که اسمش شبیه اسم پروژه ارجاعی باشه و اگر پیدا کنه، ارجاع رو عوض می‌کنه
البته برای جلوگیری از به هم ریختگی، نرم افزار از فایل‌های پروژه ای که دستکاری می‌کنه، پشتیبان می‌گیره دانلود پروژه

توجه:
* این برنامه از تمامی جهات تست نشده است، با ریسک خودتون ازش استفاده کنید (ما تو شرکت دیگه ریسکی نداریم :))
* سیستم نامگذاری اسمبلی‌ها و پروژه‌های ما ممکنه فرق کنه
* اگر به مشکلی برخوردید، لطفا زیر همین مطلب برام بنویسید
* انتخاب پوشه اسمبلی ها، الزامی نیست

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۰۲ ۱۱:۴۰

ممنون. ایده خوبی هست.

یک روش دیگر هم استفاده از نیوگت هست برای مدیریت لوکال وابستگی‌ها

Creating and then using a NuGet local repository

How to access NuGet when NuGet.org is down

نویسنده: بهزاد
تاریخ: ۱۳۹۲/۰۲/۰۲ ۱۱:۴۲

در مورد وابستگی‌های نوگت کاری نکردیم، فقط در مورد پروژه‌ها و اسمبلی هاست

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۰۲ ۱۲:۳۰

نیوگت لوکال روی شبکه می‌تونید تعریف کنید بر اساس وابستگی‌های داخلی خودتون. یعنی کاملا مستقل از نیوگت روی اینترنت.

نویسنده: یوسف نژاد
تاریخ: ۱۳۹۲/۰۲/۰۳ ۸:۲۹

شما میتونین خروجی تمام پروژه‌های ریفرنس داده شده در پروژه‌های دیگه رو به یک مسیر مشخص و مشترک تنظیم کنید. تمام پروژه‌ها هم ریفرنس خودشون رو از اون مسیر مشخص بگیرن. سپس فایلهای dll. یا exe. موردنظر رو بصورت multi-check out تنظیم کنید. بعدش هرکسی که آخرین نسخه از اون کتابخونه رو داره توسعه میده هر روز چکین کنه و بقیه هم هر روز get latest کنن. کاری که ما داریم به راحتی در شرکت خودمون انجام میدیم.

نویسنده: سیروس
تاریخ: ۱۳۹۲/۰۲/۰۳ ۱۳:۳۱

ما هم از این روش استفاده می‌کنیم.

نویسنده: منیژه محمدی
تاریخ: ۱۳۹۲/۰۸/۲۶ ۲۱:۱۳

در مورد TFS چطور ؟ در مورد ان پیشنهادی ندارید؟

عنوان: **یکپارچه سازی TortoiseSVN و YouTrack**
نویسنده: جوادی
تاریخ: ۱۳۹۲/۰۷/۱۱ ۰:۳۰
آدرس: www.dotnettips.info
برچسب‌ها: Subversion, SVN, Source control, TortoiseSVN, Integration IssueTracker

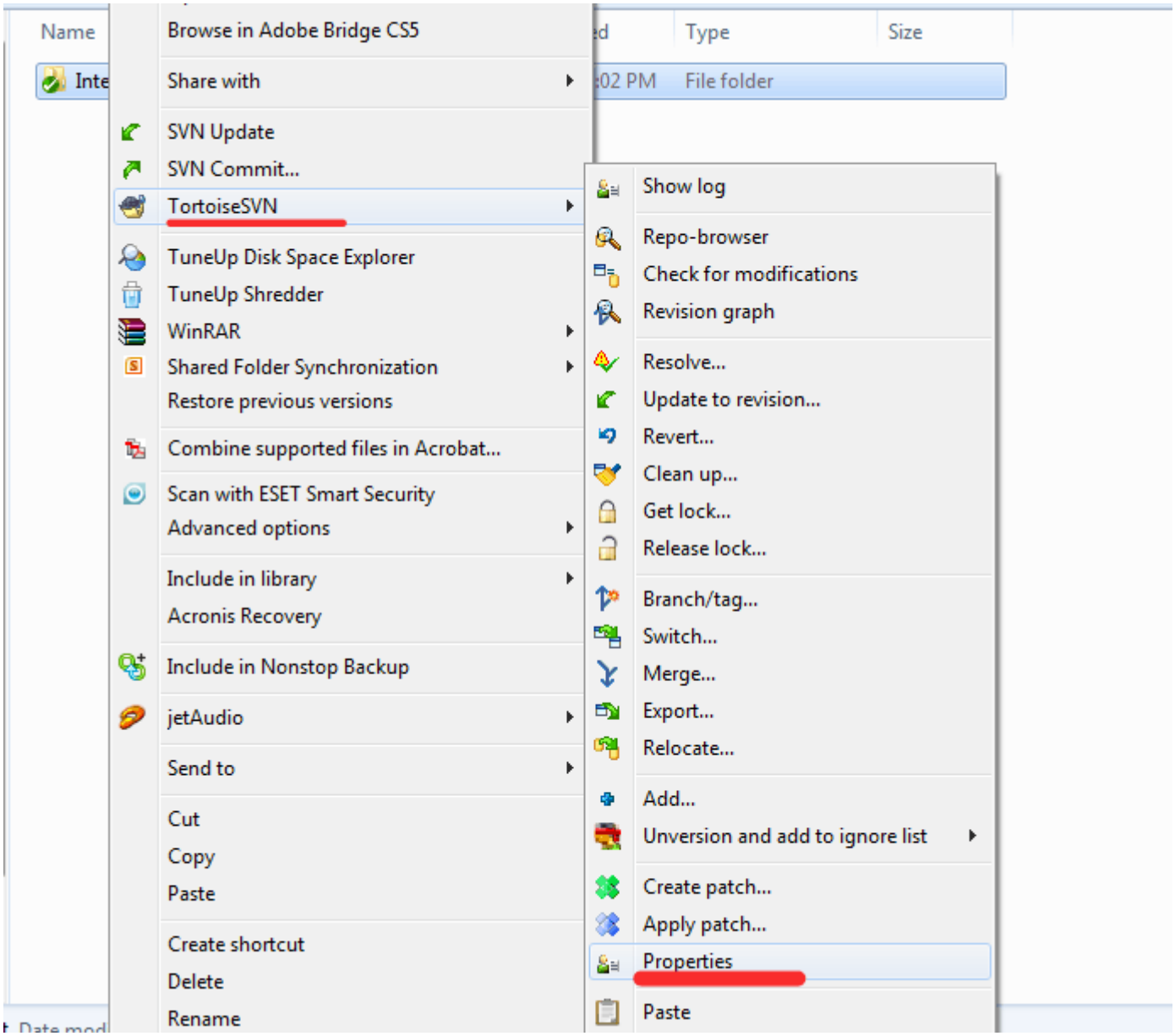
پیش نیاز اگر در مورد TortoiseSVN و سورس کنترل اطلاعات پایه ندارید، کتاب مدیریت فایلهای یک پروژه نرم افزاری با استفاده از Subversion آقای نصیری را مطالعه کنید و همچنین سیستم پیگیری خطای YouTrack را نگاهی بیاندازید (البته اگر اطلاعی ندارید) .

مقدمه

هنگام کار روی یک پروژه، باگ ها، وظیفه‌ها و موضوعاتی به شما واگذار می‌شود که باید انها را انجام دهید. هنگام commit کردن تغییرات، برای مشخص شدن اینکه تغییرات مربوط به کدام Bug-Id بوده است بود است باید سیستم Bug/Issue Tracker رو با سورس کنترل یکپارچه کنیم.

یکپارچه سازی TortoiseSVN و YouTrack

-1 روی یک نسخه کاری پروژه راست کلیک، از منوی TortoiseSVN گزینه Properties را انتخاب کنید.

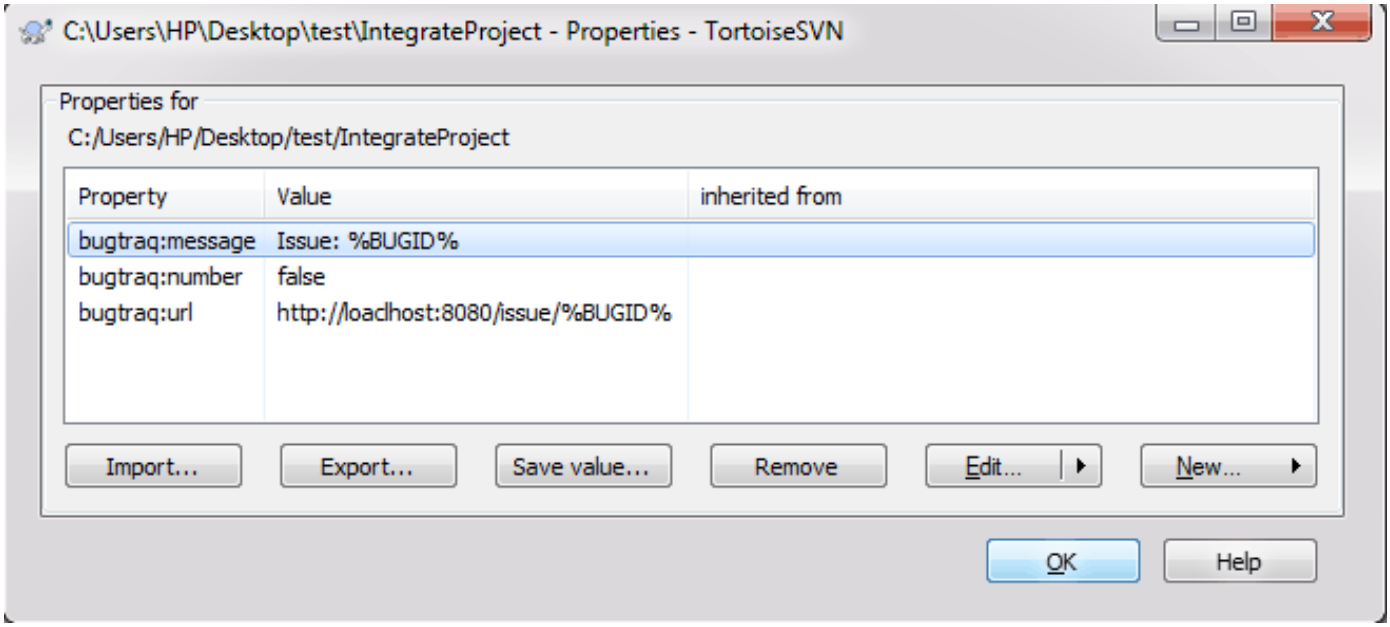


-2 از پنجر باز شده دکمه New، گزینه Other را انتخاب کنید. در پنجره باز شده از منوی کشویی مربوط به Property Name، مقادیر

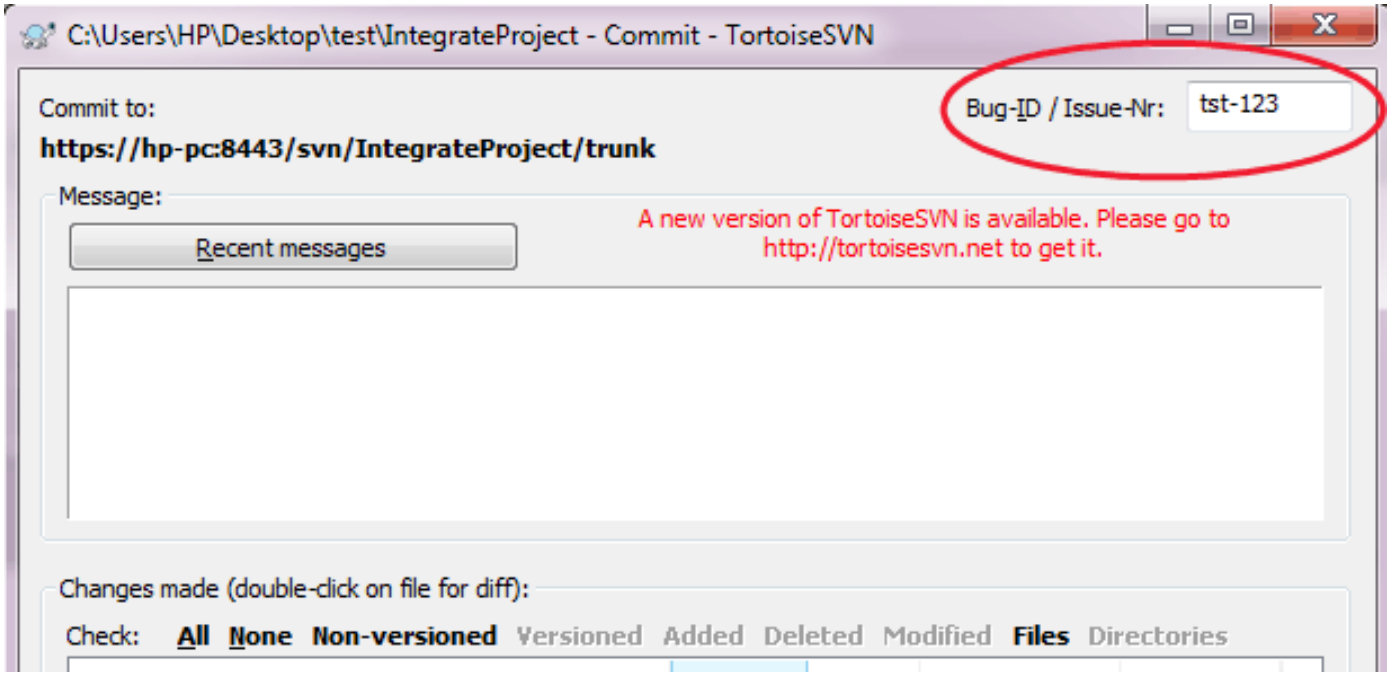
خصلت‌های زیر را تنظیم کنید: **bugtraq:url** : آدرس YouTrack Sever که به این صورت وارد می‌شود: %
http://localhost:8080/issue/%BUGID

bugtraq:message : درو اقع الگویی پیامی هست که برای نگهداری Bug-Id استفاده می‌شود و باید شامل کلمه %BUGID% باشد.
مثلا: Issue: %BUGID %

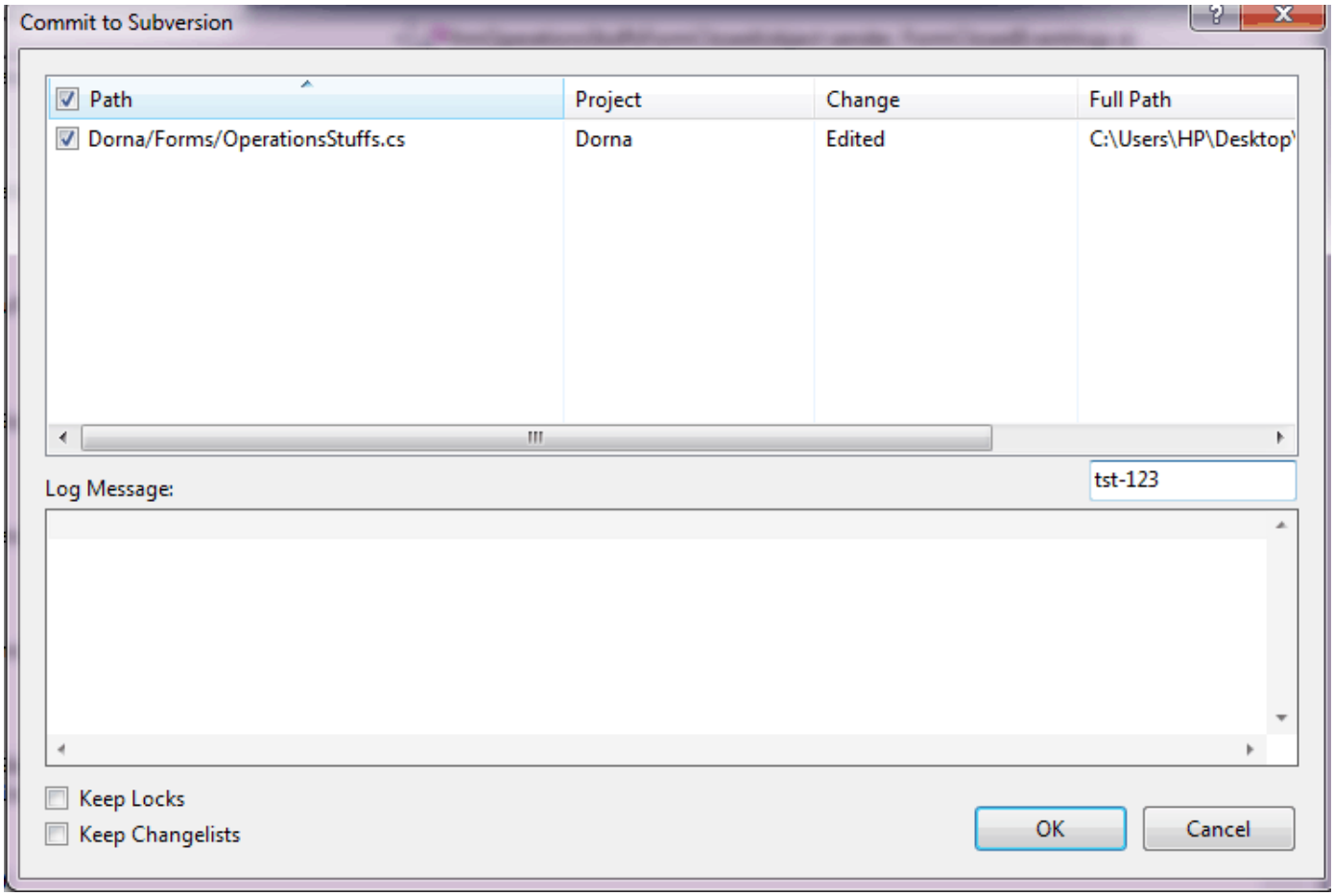
bugtraq:number : مقدار این خصلت را false وارد کنید؛ چون Bug-Idهای YouTrack می‌توانند شامل عدد و حروف باشند.



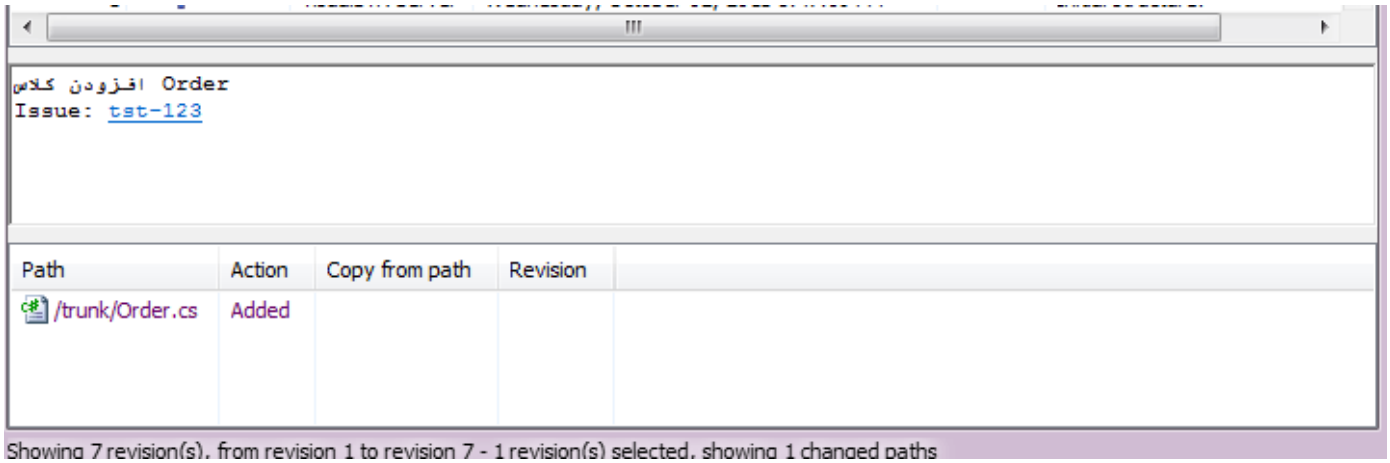
بعد از اینکه این سه خصلت را مقداردهی کرید، تغییرات را Commit کنید. همانطور که می‌بینید یک Textbox (بالا، سمت راست) اضافه شده که محل وارد کردن Bug-Id مربوط به تغییرات است. از این پس، می‌توانید Bug-Id یا Issue-Idهای مربوط به هر تغییرات را در آن Textbox وارد کنید.



همچنین تغییرات در پلاگین AnkhSVN در ویژال استودیو نیز اعمال می‌شود:



اکنون، در متن commitها شماره Bud-Id نیز ذکر شده است.

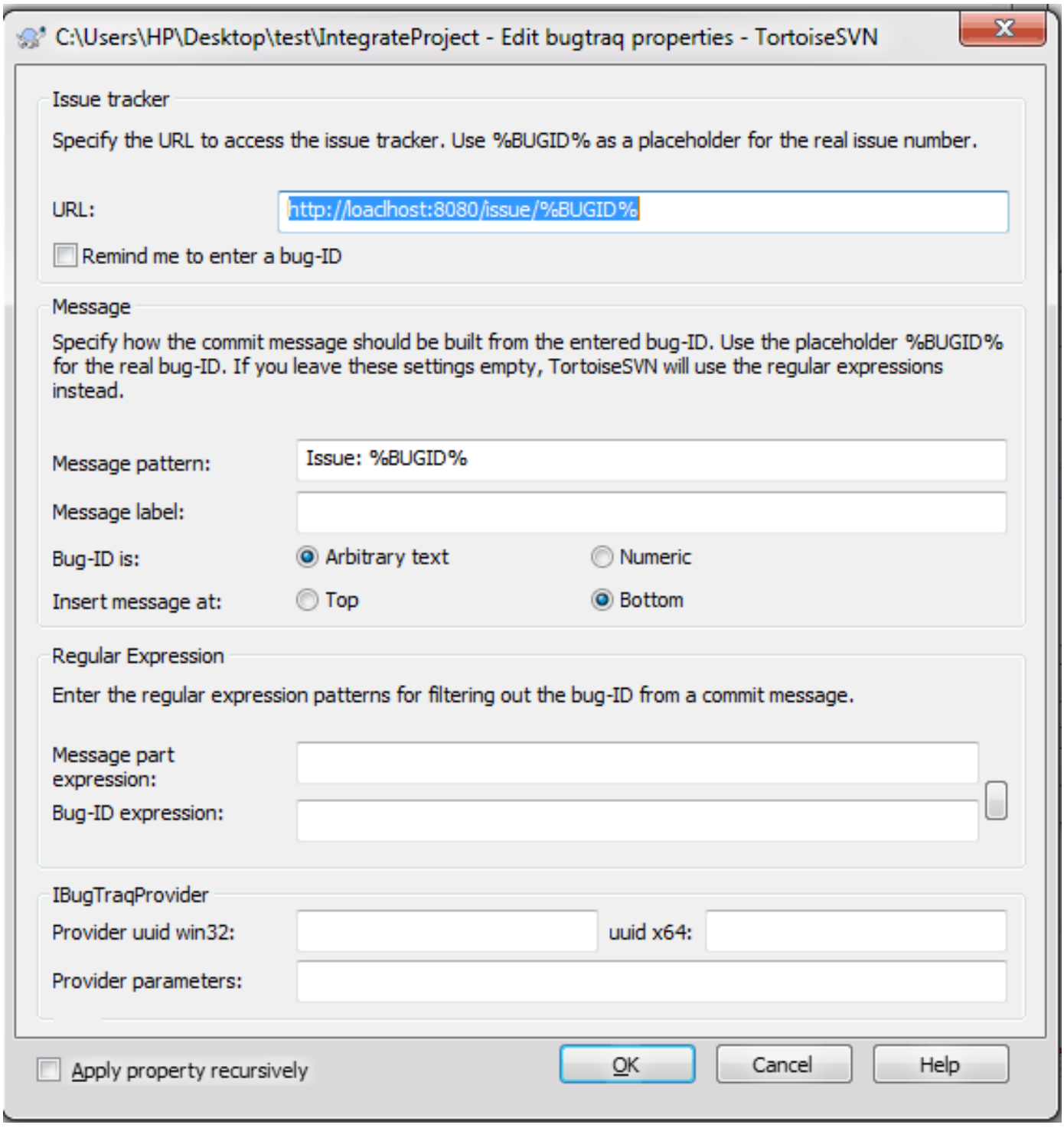


نکته 1: اگر YouTrack روی یک سرور نصب هست، بجای localhost نام کامپیوتر سرور یا آی پی آن را وارد کنید. پورت 8080

نیز بصورت پیش فرض است و اگر هنگام نصب آن را تغییر داده اید، اینجا نیز آنرا تغییر دهید.

نکته 2: خصلت bugtraq:message یک الگوی پیام از شما می‌گیرد؛ یعنی الگو را تحت هر شکلی می‌توان وارد کرد. بعنوان مثال الگو را به این شکل وارد کنید: "برای مشاهده جزئیات بیشتر به Bug-Id شماره %BUGID% مراجعه کنید." **نکته 3:** اگر خصلت bugtraq:number مقدارش true باشد، برای وارد کردن Bug-Id فقط از عدد می‌توانید استفاده کنید. بصورت پیش فرض مقدار این خصلت true است. **نکته 4:** می‌توانید این تنظیمات را در یک فایل Export کنید و در بقیه پروژه ها، با یک مرحله و بسادگی آنرا Import کنید.

خصلت‌های دیگری نیز می‌توان برروی مخزن کد اعمال کرد که از حوزه این مقاله خارج است. همچنین تنظیمات اختیاری جانبی دیگری نیز برای یکپارچه سازی وجود دارند. برای دیدن این تنظیمات روی نسخه کاری راست کلیک، از منوی TortoiseSVN گزینه Properties را انتخاب کنید و از پنجره باز شده روی دکمه New و گزینه (Issue tracker integration) Bugtraq را انتخاب کنید.



برای اطلاعات بیشتر در مورد این تنظیمات، داکیومنت یکپارچه سازی با سیستم‌های Bug tracking / Issue Tracking را مطالعه کنید.

عنوان: **مفاهیم پایه سیستم های کنترل نسخه؛ قسمت اول : گیت**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۵/۱۳ ۱۴:۴۵
آدرس: www.dotnettips.info
گروه‌ها: Git, SVN, Source control, GitHub

در این مقاله با دو سیستم کنترل نسخه git و SVN آشنا شده و تفاوت‌های آن‌ها را برای تازه‌کاران بررسی می‌کنیم. ایده اولیه نوشتن این مقاله زمانی بود که برای یک پروژه‌ای، اعضای تیم ما دور هم جمع شده و در مورد ابزارهای مورد استفاده بحث کردند و یک عده از گیت و عده‌ای از SVN صحبت می‌کردند. بر این شدم که مقاله‌ای نوشته و ابتدا به معرفی آن‌ها و سپس به مزایا و معایب هر کدام بپردازیم.

امروزه، استفاده از سیستم‌های کنترل نسخه (*Version Control System*) رواج زیادی پیدا کرده است. این سیستم‌ها به شما اجازه می‌دهند تا تغییراتی را که در پروژه ایجاد می‌شوند، ضبط و ثبت کرده تا از تغییراتی که در سطح پروژه اتفاق می‌افتد آگاه شوید. با ذکر یک نمونه این تعریف را باز میکنم:

شما به صورت تیمی در حال انجام یک پروژه هستید و باید نسبت به تغییراتی که اعضای تیم در یک پروژه می‌دهند، آگاه شوید. هر برنامه نویس بعد از انجام تغییرات باید این تغییرات را در سیستم کنترل نسخه به روز کند تا بتوان به سوالات زیر پاسخ داد: آیا اگر در بین راه به مشکل برخوردید می‌توانید پروژه خود را به یک یا چند گام عقب‌تر برگردانید؟ آیا می‌توانید به هر یک از اعضاء تیم دسترسی‌هایی را به قسمت هایی از پروژه تعیین کنید؟ می‌توانید تفاوت فایل‌های تغییر یافته را بیابید؟ آیا میتوان خطاهای یک برنامه را گزارش داد و به بحث در مورد آن پرداخت؟ چه کسی کدها را تغییر داده است؟ روند کار و تغییرات به چه صورت است؟ (این مورد برای به روز کردن نمودارهای burndown در توسعه چابک می‌تواند بسیار مفید باشد.)

پی نوشت: نه تنها در یک تیم بلکه بهتر هست در یک کار انفرادی هم از این سیستم‌ها استفاده کرد تا حداقل بازبینی روی پروژه‌های شخصی خود هم داشته باشیم.

سیستم کنترل گیت: این سیستم در سال 2005 توسط لینوس تورووالدز خالق لینوکس معرفی شد و از آن زمان تاکنون یکی از پر استفاده‌ترین سیستم‌های کنترل نسخه شناخته شده است. ویکی پدیا گیت را به این شکل تعریف می‌کند: « *یک سیستم بازبینی توزیع شده با تاکید بر جامعیت داده‌ها، سرعت و پشتیبانی جهت توزیع کار.* »

از معروف‌ترین سیستم‌های هاستینگ که از گیت استفاده می‌کنند، می‌توان به گیت هاب اشاره کرد.

اکثر سیستم‌های هاستینگ گیت، دو حالت را ارائه می‌دهند: عمومی : در این حالت کدهای شما به عموم بازدیدکنندگان نمایش داده می‌شود و دیگران هم می‌توانند در تکمیل و ویرایش کدهای شما مشارکت کنند و این امکان به صورت رایگان فراهم است. سیستم گیت هاب به دلیل محبوبیت زیادی که دارد، در اکثر اوقات انتخاب اول همه کاربران است. خصوصی: در این حالت کد متعلق به شما، یا شرکت یا تیم نرم افزاری شماست و غیر از افراد تعیین شده، شخص دیگری به کدهای شما دسترسی ندارد. اکثر سیستم‌های مدیریتی این مورد را به صورت premium پشتیبانی می‌کنند. به این معنا که باید اجاره آن را به طور ماهانه پرداخت کنید. سیستم گیت هاب ماهی پنج دلار بابت آن دریافت می‌کند. سیستم دیگری که در این زمینه محبوبیت دارد سیستم BitBucket هست که که اگر تیم شما کوچک است و در نهایت پنج نفر هستید، می‌توانید از حالت خصوصی به طور رایگان استفاده کنید ولی اگر اعضای تیم شما بیشتر شد، باید هزینه‌ب اجاره آن را که از 10 دلار آغاز می‌گردد، به طور ماهیانه پرداخت کنید.

پی نوشت: میتوانید از سیستم‌های متن باز رایگان هم که قابل نصب بر روی هاست ها هم هستند استفاده کنید که در این حالت تنها هزینه هاست یا سرور برای شما می‌ماند.

در سیستم گیت اصطلاحات زیادی وجود دارد: **Repository یا مخزن: برای** هر پروژه‌ای که ایجاد می‌شود، ابتدا یک مخزن ایجاد شده و کدها داخل آن قرار می‌گیرند. کاربرانی که قصد تغییر پروژه را دارند باید یک مخزن جداگانه ایجاد کنند تا بعدا تمامی تغییرات آن‌ها را روی پروژه‌ی اصلی اعمال کنند.

Fork: هر کاربری که قصد تغییر را بر روی سورس کدی، داشته باشد، ابتدا باید پروژه‌ی نویسنده اصلی پروژه را به یک مخزنی که متعلق به خودش هست انتقال دهد. به این عمل Fork کردن می‌گویند. حال کاربر تغییرات خودش را اعمال کرده و لازم هست که این تغییرات با پروژه‌ی اصلی که به آن Master می‌گوییم ادغام شوند. بدین جهت کاربر فرمان pull request را می‌دهد تا به نویسنده‌ی اصلی پروژه این موضوع اطلاع داده شود و نویسنده‌ی اصلی در صورت صلاحدید خود آن را تایید کند.

Branching یا شاخه بندی: نویسنده‌ی مخزن اصلی می‌تواند با مفهومی با نام شاخه بندی کار کند. او با استفاده از این مفهوم، پروژه را به قسمت یا شاخه‌های مختلف تقسیم کرده و همچنین با ایجاد دسترسی‌های مختلف به کاربران اجازه تغییرات را بدهد. به عنوان مثال بخش‌های مختلف پروژه از قبیل بخش منطق برنامه، داده ها، رابط کاربری و ... می‌تواند باشد. بعد از انجام تغییرات روی یک شاخه می‌توانید درخواست merge ادغام شدن یا کل پروژه را داشته باشید. در عمل شاخه بندی، هیچ کدام از شاخه‌های بر

روی یک دیگر تاثیر یا دخالتی ندارند و حتی می‌توانید چند شاخه را جدا از بخش master با یکدیگر ادغام کنید.

به غیر از ارتباط خط فرمانی که میتوان با گیت هاب برقرار کرد، میتوان از یک سری ابزار گرافیکی خارجی هم جهت ایجاد این ارتباط، استفاده کرد: GitHub For Windows : نسخه‌ی رسمی است که از طرف خود گیت هاب تهیه گردیده است و استفاده از آن بسیار راحت است. البته یک مشکل کوچک در دانلود آن وجود دارد که دانلود آن از طریق یک برنامه‌ی جداگانه صورت گرفته و اصلا سرعت خوبی جهت دانلود ندارد. Visual Studio .Net : (+) خود ویژوال استودیو شامل سیستمی به اسم Microsoft Git Provider است که در بخش تنظیمات می‌توانید آن را فعال کنید (به طور پیش فرض فعال است) و به هر نوع سیستم گیتی می‌توانید متصل شوید. تنها لازم است که آدرس Url گیت را وارد کنید. SourceTree : از آن دست برنامه‌های محبوبی است که استفاده آسانی دارد و خودم به شخصه از آن استفاده می‌کنم. شامل دو نسخه‌ی ویندوز و مک است و میتوانید با چندین سیستم گیت مثل «گیت هاب» و «بیت باکت» که در بالا به آن‌ها اشاره شد، به طور همزمان کار کند.

نظرات خوانندگان

نویسنده: سید محمد حسین موسوی
تاریخ: ۱۳۹۴/۰۵/۱۴ ۰:۵۱

سلام؛ خیلی ممنون. چندتا سوال :
«پی نوشت: نه تنها در یک تیم بلکه بهتر هست در یک کار انفرادی هم از این سیستم‌ها استفاده کرد تا حداقل بازبینی روی پروژه‌های شخصی خود هم داشته باشیم.»
-1این یعنی اینکه اگر من بخوام برای خودم هم به تنهایی استفاده کنم و خصوصی هم باشه باید پول بدم؟ حالا اگر عمومی باشه می‌تونم به هیچ کس اجازه دستری ندم؟ فرق عمومی که اجازه دسترسی ندی با خصوصی تو چیه؟ دیدن و ندیدن کدها ؟
2-team foundation ماکروسافت هم برای اینکارهاست؟
-3می شه کمی بیشتر در این مورد توضیح بدید؟
«پی نوشت: میتوانید از سیستم‌های متن باز رایگان هم که قابل نصب بر روی هاست‌ها هم هستند استفاده کنید که در این حالت تنها هزینه هاست یا سرور برای شما می‌ماند.»

نویسنده: محسن خان
تاریخ: ۱۳۹۴/۰۵/۱۴ ۱:۹

بحث git با هاست‌های عمومی git مثل github متفاوت هست. شما خودت هم می‌تونی یک هاست git راه اندازی کنی: راه اندازی سرور Git با استفاده از Bonobo Git Server و انتقال از ساب ورژن به گیت

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۵/۱۴ ۱:۳۰

مبحث TFS کاملا با مباحث سیستم‌های کنترل نسخه متفاوت است و یک سیستم ALM به حساب میاد نه VCS

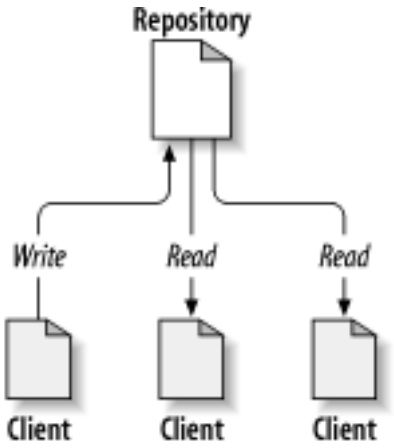
فرقی نمی‌کند، پروژه عمومی همیشه نمایش داده می‌شود، این دسترسی‌ها مربوط به شاخه بندی پروژه است که چه کسانی بتوانند تا چه حدی روی هر شاخه تغییرات را اعمال کنند ولی بحث خصوصی سازی نیاز به پرداخت هزینه دارد. هنگامی که در گیت هاب پروژه خودتون رو به صورت عمومی انتخاب کنید هیچ گزینه اضافی ندارد ولی وقتی روی خصوصی تنظیم کنید با مجموعه‌ای از آیکن‌های کارت‌های اعتباری روبرو می‌شوید.

همینطور که دوست عزیزمان "محسن خان" گفتند شما میتوانید از طریق یک سیستم متن باز و رایگان به ایجاد یک سیستم گیت جداگانه (شخصی)اقدام کنید و تنها لازم است هزینه هاستی که خریدید را به سرویس دهنده هاست پرداخت کنید.

عنوان: **مفاهیم پایه سیستم های کنترل نسخه؛ قسمت دوم : SVN**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۵/۱۴ ۹:۴۵
آدرس: www.dotnettips.info
گروه‌ها: Git, SVN, Source control

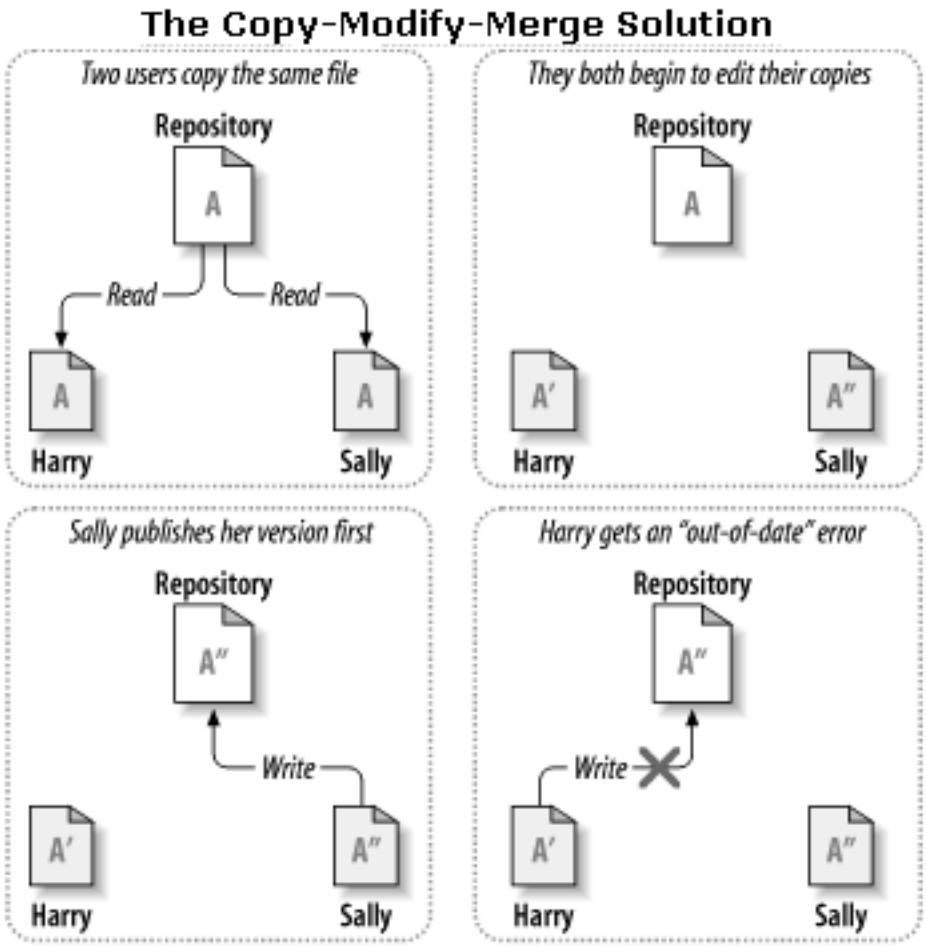
در قسمت قبلی ، اهمیت استفاده از سیستم‌های کنترل نسخه را بیان کردیم و مفاهیم پایه‌ای گیت را مورد بررسی قرار دادیم. در این قسمت مفاهیم پایه‌ای SVN را مورد بررسی قرار می‌دهیم. **SVN** مخفف عبارت SubVersion هست و یک سیستم کنترل نسخه‌ی رایگان و متن باز است که توسط شرکت کلاب نت حمایت می‌شود. به تعدادی از این سیستم‌ها، سیستم‌های «مدیریت پیکربندی نرم افزار» *Software Configuration Manager (SCM)* هم اطلاق می‌شود.

در این سیستم فایل‌ها در یک مخزن Repository مرکزی ذخیره می‌شوند و با هر تغییری که در فایل‌ها و دایرکتوری‌ها ایجاد می‌شود، آن‌ها را ثبت می‌کند. این امکان به ما این اجازه را می‌دهد که نسخه‌ی قدیمی فایل‌ها را بازیابی کرده و تاریخچه‌ی اینکه فایل‌ها چگونه و چه موقع و توسط چه کسی تغییر کرده‌اند، به ما نشان دهد. تصویر سلسله مراتبی زیر به خوبی نحوه ارتباط کلاینت‌ها را به این مخزن نشان می‌دهد.



SVN برای مدیریت چندین نسخه از فایل ها، از مدل «کپی، ویرایش، ادغام» **Copy-Modify-Merge** استفاده می‌کند. در این مدل که هر کاربری در مخزن عمل خواندن را انجام می‌دهد، یک کپی جداگانه و کاملا شخصی برای او گرفته شده و سپس کپی‌های شخصی خودش را ویرایش می‌کند. بعد از اینکه ویرایش تکمیل شد، کپی شخصی خودش را با یک فایل جدید و نهایی ادغام می‌کند.

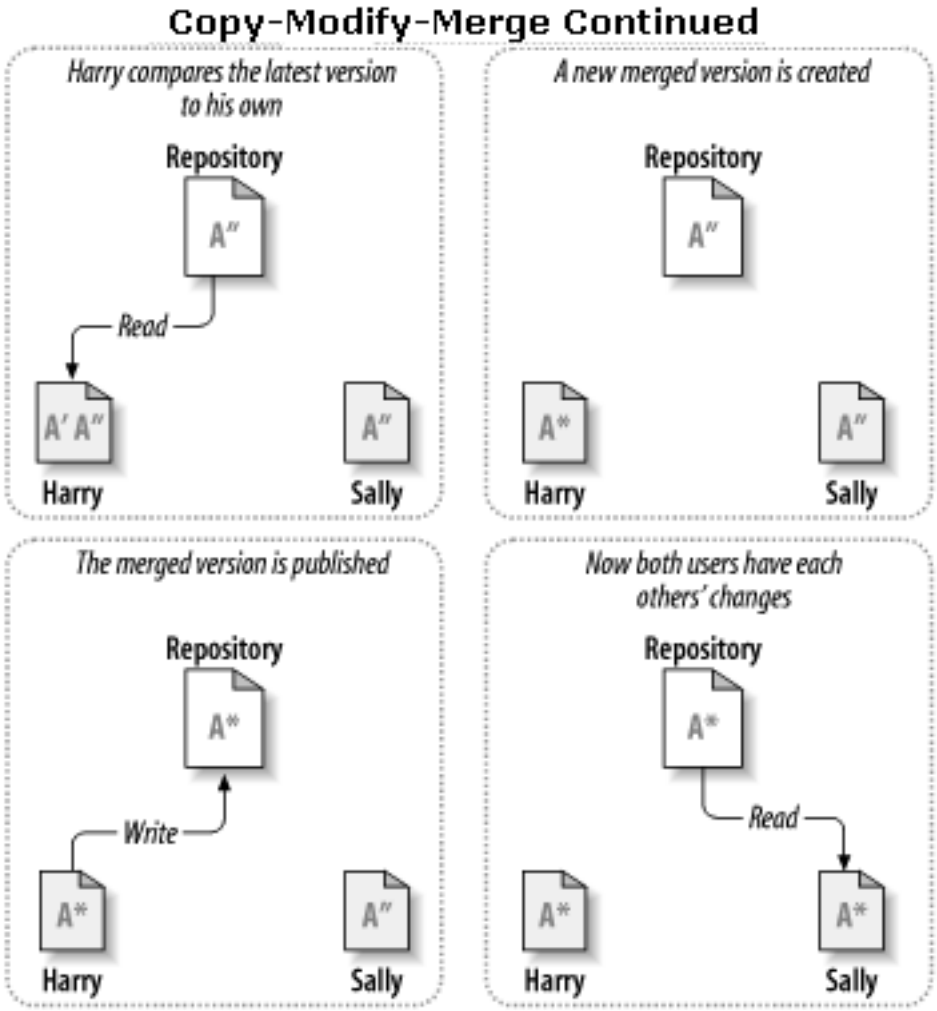
این روش به شدت از روش «قفل کردن، ویرایش، آزادسازی» » **Lock-Modify-Unlock** کارآمدتر است و دیگر نیازی نیست که یک کاربر در یک زمان به این ساختار دسترسی داشته باشد و آن را ویرایش کند.



در تصویر بالا هری و سالی، یک کپی از مخزن موجود را گرفته و سپس هر کدام جداگانه بر روی کپی‌های خودشان مشغول به کار می‌شوند. سپس سالی کارش را زودتر به اتمام رسانده و مخزن را به روز می‌کند. بعد از آن، هری هم کارش به پایان می‌رسد و قصد به روز کردن مخزن را دارد ولی سیستم به او اجازه این کار را نمی‌دهد؛ چون این مخزن آن مخزن نیست که هری قبلا از آن کپی گرفته است. آن مخزن بعد از به روزرسانی سالی تغییر یافته است. پس او مجبور است تا تغییرات جدید مخزن را دریافت کرده و کپی خودش را به روز کند. پس از آن می‌تواند کپی خودش را بر روی مخزن اعمال کند (با فرض اینکه تغییرات جدید هیچ تصادمی با تغییراتی که روی کپی خودش اعمال کرده است ندارند).

سناریو بالا با احتساب وجود تصادم

اگر همین سناریوی بالا را فرض کنیم که تغییراتی که هری روی فایل‌ها داده است همان تغییراتی است که سالی قبلا روی مخزن اصلی روی همان فایل‌ها اعمال کرده است، آیا در این حالت دریافت به روزرسانی‌های جدید باعث ایجاد تصادم می‌شود؟



هری درخواست ادغام آخرین تغییرات مخزن را با کپی خودش می‌کند. از آنجا که فایل A تصادم دارد یک فلگ *flag* از این وضعیت می‌گیرد. حال هری میتواند تفاوت‌های ایجاد شده را ببیند و بین آن‌ها یکی را انتخاب کند. در این وضعیت هری همپوشانی‌های کدها را برطرف می‌کند و شاید هم بحثی در مورد این تصادم با سالی داشته باشد تا بهترین تغییر کد انتخاب گردد و نهایتا به روشی کاملا امن و مطمئن، با مخزن اصلی ادغام می‌شود.

پی نوشت : نرم افزارها نمی‌توانند موضوع تصادم را به طور خودکار اعمال کنند. از آنجا که نیاز به تصمیم گیری و درک هوشمند دارد این کار به صورت انسانی باید بررسی گردد.

عنوان: **مفاهیم پایه سیستم های کنترل نسخه؛ قسمت سوم : جمع بندی**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۵/۱۸ ۲۱:۲۰
آدرس: www.dotnettips.info
گروه‌ها: Git, SVN, Source control, GitHub

در اولین قسمت این سری، گیت و در قسمت دوم ، SVN را بررسی کردیم؛ در این مقاله قصد داریم یک جمع بندی از این دو مقاله داشته باشیم.

احتمالا در مورد این دو سیستم حرف‌های زیادی شنیده‌اید و احتمالا بیشتر آن‌ها در مورد گیت نظر مساعدتری داشته‌اند؛ ولی تفاوت‌هایی بین این دو سیستم هست که باید به نسبت هدف و نیازی که دارید آن را مشخص کنید. یکی از اصلی‌ترین این تفاوت‌ها این است که svn یک سیستم مرکزی است؛ ولی گیت اینگونه نیست که در ادامه تفاوت این دو مورد را تشریح می‌کنیم.

یک. SVN یک مخزن مرکزی دارد که همه‌ی تغییراتی که روی کپی‌ها انجام می‌شود، باید به سمت مخزن مرکزی Commit یا ارسال شوند. ولی در سیستم گیت یک سیستم مرکزی وجود ندارد و هر مخزنی که fork یا Clone می‌شود، یک مخزن جداگانه به حساب می‌آید و Commit شدن تنها به مخزن کپی شده صورت میگیرد و در صورت pull request ادغام با مخزن اولیه خودش صورت میگیرد. دو. گیت به نسبت svn از پیچیدگی بیشتری برخوردار است؛ ولی برای پروژه‌های بزرگتر که کاربران زیادی با آن کار می‌کنند و احتمال شاخه بندی‌های زیادتر، در آن وجود دارد بهتر عمل می‌کند. موقعی که یک پروژه یا تیم کوچکی روی آن کار می‌کنند به دلیل commit شدن مستقیمی که svn دارد، کار راحت‌تر و آسان‌تر صورت می‌گیرد ولی با زیاد شدن کاربران و حجم کار، گیت کارآیی بالاتری دارد. سه. از آن جا که گیت نیاز به fork شدن دارد و یک مخزن کاملا مجزا از پروژه اصلی تولید می‌کند؛ سرعت بهتری نسبت به svn که یک کپی از زیر مجموعه ساختار اصلی ایجاد می‌کند دارد. *چهار*. شاخه بندی یک مفهوم اصلی و مهم در گیت به شمار می‌آید که اکثر کاربران همه روزه از آن استفاده می‌کنند و این اجازه را می‌دهد که که تغییرات و تاریخچه فعالیت هر کاربر را بر روی هر شاخه، جداگانه ببینیم. در svn پیاده سازی شاخه‌ها یا تگ‌ها سخت و مشکل است. همچنین شاخه بندی کار در svn به شکل سابق با کپی کردن صورت گرفته که گاهی اوقات به دلایلی که در قسمت قبل گفتیم، باعث ناسازگاری می‌گردد. پنج. حجم مخازن گیت به نسبت svn خیلی کمتر است برای نمونه پروژه موزیلا 30 درصد حجم کمتری در مخزن گیت دارد. یکی از دلایلی که svn حجم بیشتری میگیرد این است که به ازای هر فایل دو فایل موجود است یکی که همان فایل اصلی است که کاربر با آن کار می‌کند و دیگری یک فایل دیگر در شاخه svn. است که برای کمک به عملیاتی چون وضعیت، تفاوت ها، ثبت تغییرات به کار می‌رود. در صورتی که در آن سمت، گیت، تنها به یک فایل شاخص 100 بایتی برای هر دایرکتوری کاری نیاز دارد *شش*. گیت عملیات کاربری را به جز fetch و push، خیلی سریع انجام میدهد. این عملیات شامل یافتن تفاوت‌ها، نمایش تاریخچه، ثبت تغییرات، ادغام شاخه‌ها و جابجایی بین شاخه‌ها می‌گردد. *هفت*. در سیستم SVN به دلیل ساختار درختی که دارد، می‌توانید زیر مجموعه‌ی یک مخزن را بررسی کنید ولی در سیستم گیت اینکار امکان پذیر نیست. البته باید به این نکته توجه داشت که برای یک پروژه‌ی بزرگ شما مجبور هستید همیشه کل مخزن را دانلود کنید. حتی اگر تنها نسخه‌ی خاصی از این زیرمجموعه را در نظر داشته باشید. به همین علت در شهرهایی که اینترنت گرانقیمت و یا سرعت پایین عرضه می‌شود، گیت به صرفه‌تر است و زمان کمتری برای دانلود آن می برد. **موارد تعریف شده زیر طبق گفته ویکی سایت Kernel.Org ذکر می‌شود:**

گیت از سیستم SVN سریعتر عمل می‌کند.

در سیستم گیت هر شاخه بندی کل تاریخچه خود را به دنبال دارد.

فایل git که تنظیمات مخزن داخلش قرار دارد، ساختار ساده‌ای دارد و به راحتی می‌توان در صورت ایجاد مشکل، آن را حل کرد و به ندرت هم پیش می‌آید که مشکلی برایش پیش بیاید.

پشتیبانی گیری از یک سیستم مرکزی مثل SVN راحت‌تر از پشتیبانی گیری از پوشه‌های توزیع شده در مخزن گیت است.

ابزارهای کاربری svn تا به الان پیشرفت‌های چشمگیری داشته است. پلاگین‌ها و برنامه‌های بیشتری نسبت به سیستم گیت دارد. یکی از معروفترین این پلاگین‌ها، ابزار tortoisesvn است (البته ابزارهای گیت امروز رشد چشمگیرتری داشته اند که در قسمت اول نمونه‌های آن ذکر شد).

سیستم svn برای نسخه بندی و تشخیص تفاوت‌ها از یک سیستم ساده اعداد ترتیبی استفاده می‌کند که اولین ثبت با شماره یک آغاز شده و به ترتیب ادامه می‌یابد و برای کاربران هم خواندنش راحت است و هم قابل پیش بینی است. به همین جهت برای بررسی تاریخچه‌ها و دیگر گزارش‌ها تا حدی راحت عمل می‌کند. در سیستم شاخه بندی این سیستم شماره گذاری چندان مطلوب نیست و متوجه نمی‌شوید که این شاخه از کجا نشات گرفته است. در حال حاضر برای پروژه‌ی موزیلا این عدد به 6 رقم رسیده است ولی در آن سمت، سیستم گیت از هش SH-1 استفاده می‌کند که یک رشته 40 کاراکتری است و 8 رقم اول آن به منشاء اشاره می‌کند که باعث می‌شود متوجه بشویم که این شاخه از کجا آمده است ولی از آنجا که این عدد یکتا ترتیبی نیست، برای خواندن و

گزارشگیری‌هایی که در SVN راحت صورت می‌گیرد، در گیت ممکن نیست یا مشکل است.
گیت رویدادهای ادغام و شاخه بندی را بهتر انجام می‌دهد.

نظرات خوانندگان

نویسنده: محمدرضا کنی
تاریخ: ۱۳۹۴/۰۷/۱۴ ۱۰:۳۸

با سلام و تشکر از مقاله
من قبلا با svn کار کردم ولی الان متوجه شدم git برای من بهتر هستش
آیا ابزاری که محیط گرافیکی داشته باشه مثل tortoisesvn برای git هم وجود داره؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۷/۱۴ ۱۱:۴۵

- « ساده‌ترین روش کار با Github در ویندوز »
- « مراحل ارسال یک پروژه‌ی Visual Studio به GitHub »

و ...