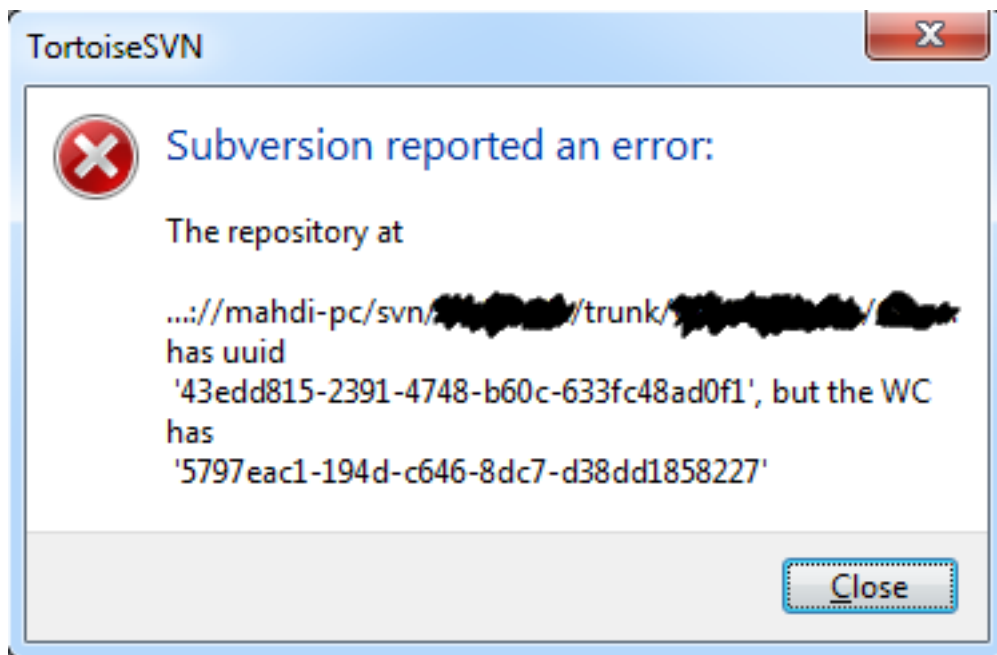
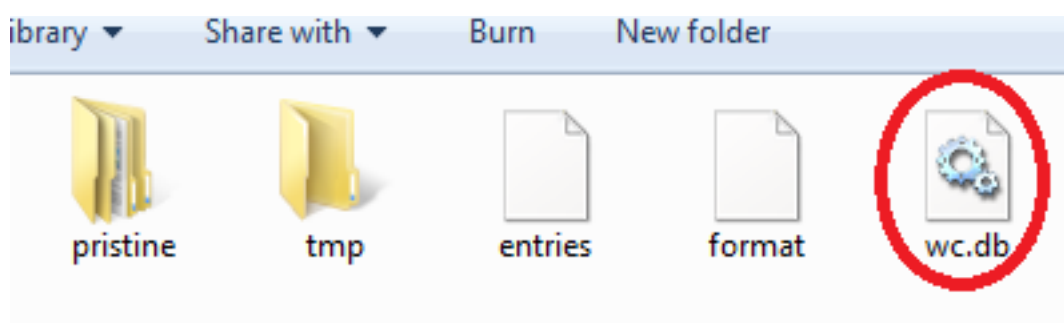


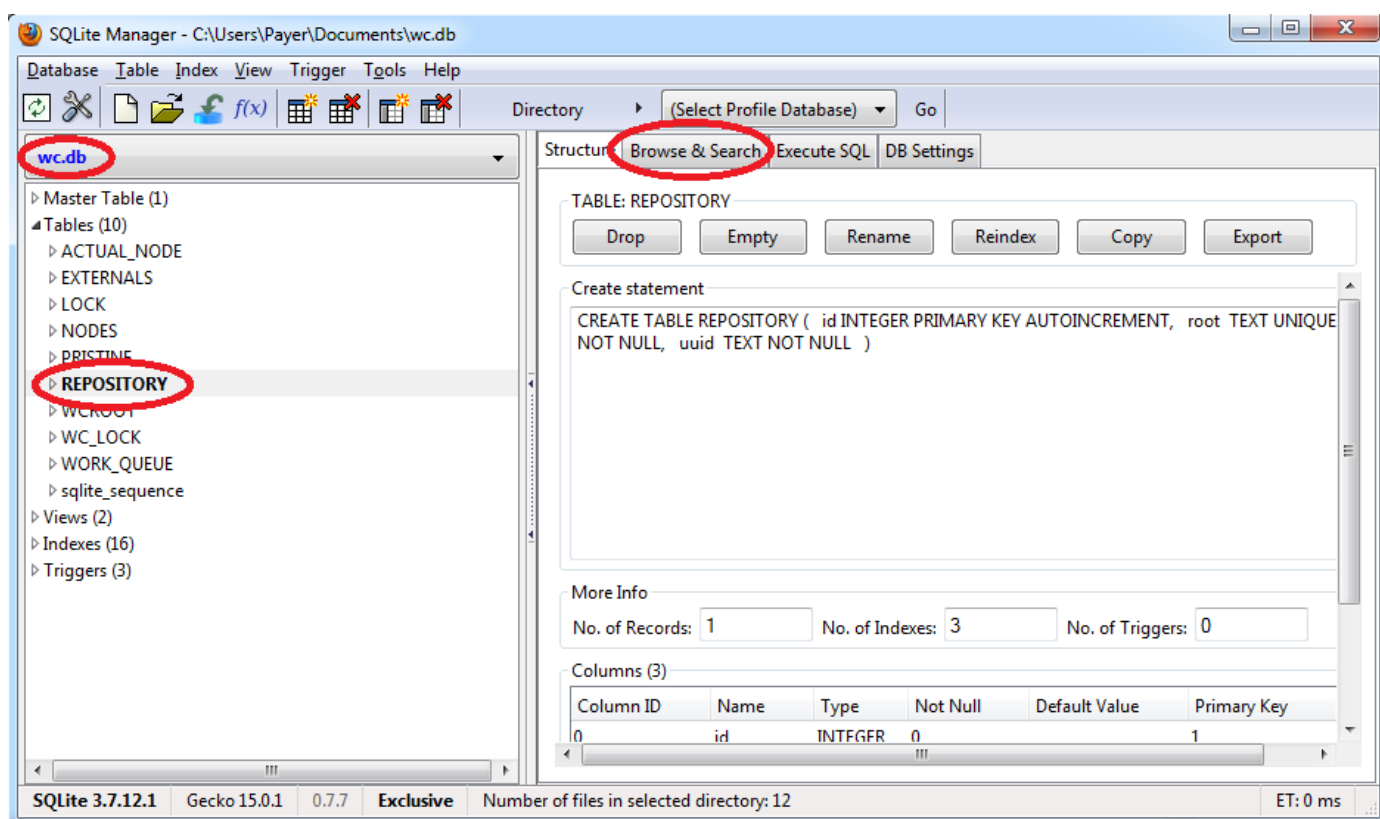
بنا به دلایلی، کدهای نگهداری شده توسط svn در سیستم خودم رو باید به سیستم دیگری وارد می‌کردم که شامل چند پروژه می‌شد و در قسمت مربوطه چند پروژه کار شده بود. تا این که مشکلی پیش آمد و جهت ادامه توسعه مجبور به برگرداندن پروژه‌ها به سیستم قبلی شدیم. حالا مشکل این بود که به repository قبلی هیچ گونه دسترسی 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:



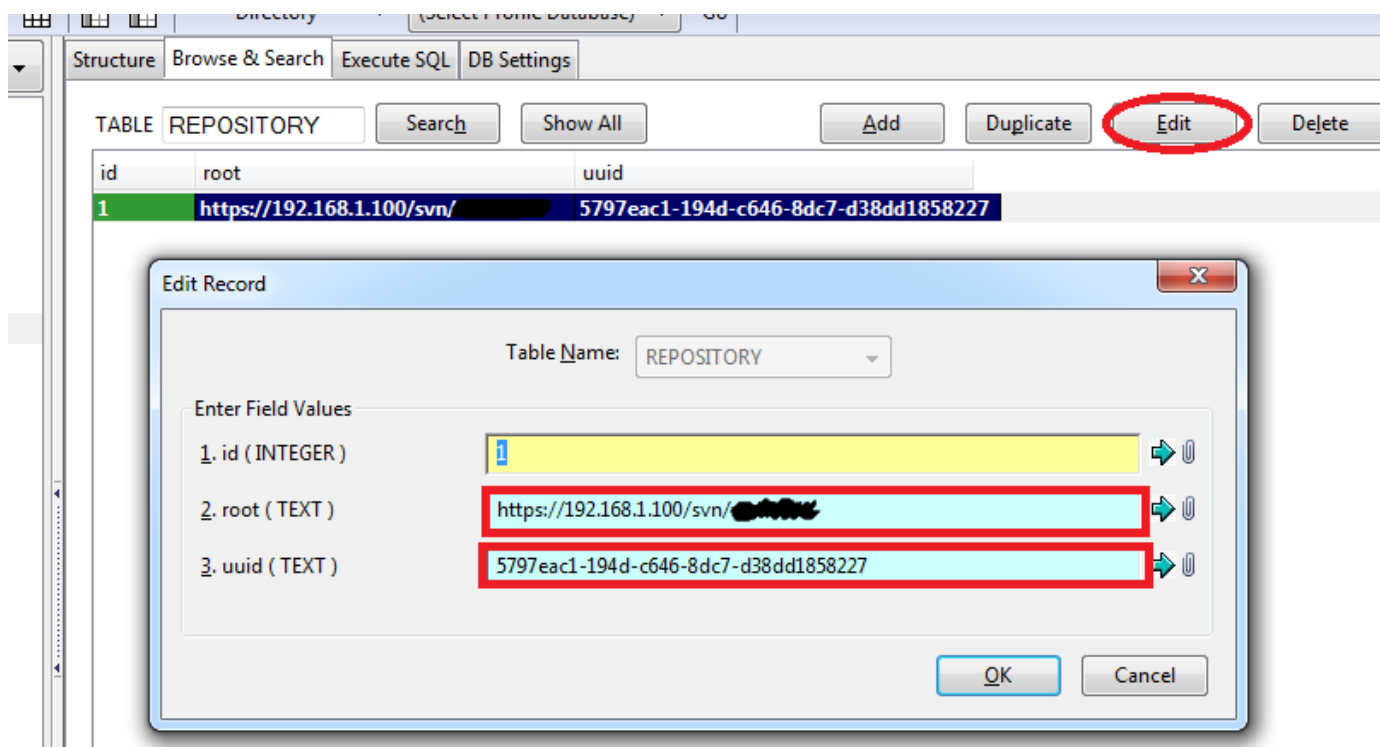
بله؛ مشکل از اینجا ناشی می‌شد که سرور دوم کدها را import نکرده بود و بلکه با یک کامیت ساده، کدها وارد سرور شده بودند.
راه حل: در کل برای تغییر این uid که در تصویر مشخص شده، باید فایل SQLite مربوطه را یافت و به راحتی، دو مشخصه مربوط به آدرس مخزن و uid را در آن اصلاح کرد.
 در پوشه svn کنار پروژه، فایلی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



که به سادگی می‌توان آن را 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 پروژه ها در کنار هم نیز دقت داشت. مثلا به محل آخری که در آدرس وجود دارد باید دقت کرد تا در مسيردهی، 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.

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 مواجهه است. چرا که برخی از پروژه‌های ارجاعی، روی کامپیوتر برنامه نویس 1 وجود نداره. برعکسش 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: ۱۱:۴۰ ۱۳۹۲/۰۲/۰۲

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: ۱۱:۴۲ ۱۳۹۲/۰۲/۰۲

در مورد وابستگی‌های نوگت کاری نکردیم، فقط در مورد پروژه‌ها و اسمبلی‌هاست

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

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

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

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: ۱۳:۳۱ ۱۳۹۲/۰۲/۰۳

ما هم از این روش استفاده می‌کنیم.

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

در مورد TFS چطور ؟ در مورد ان پیشنهادی ندارید؟

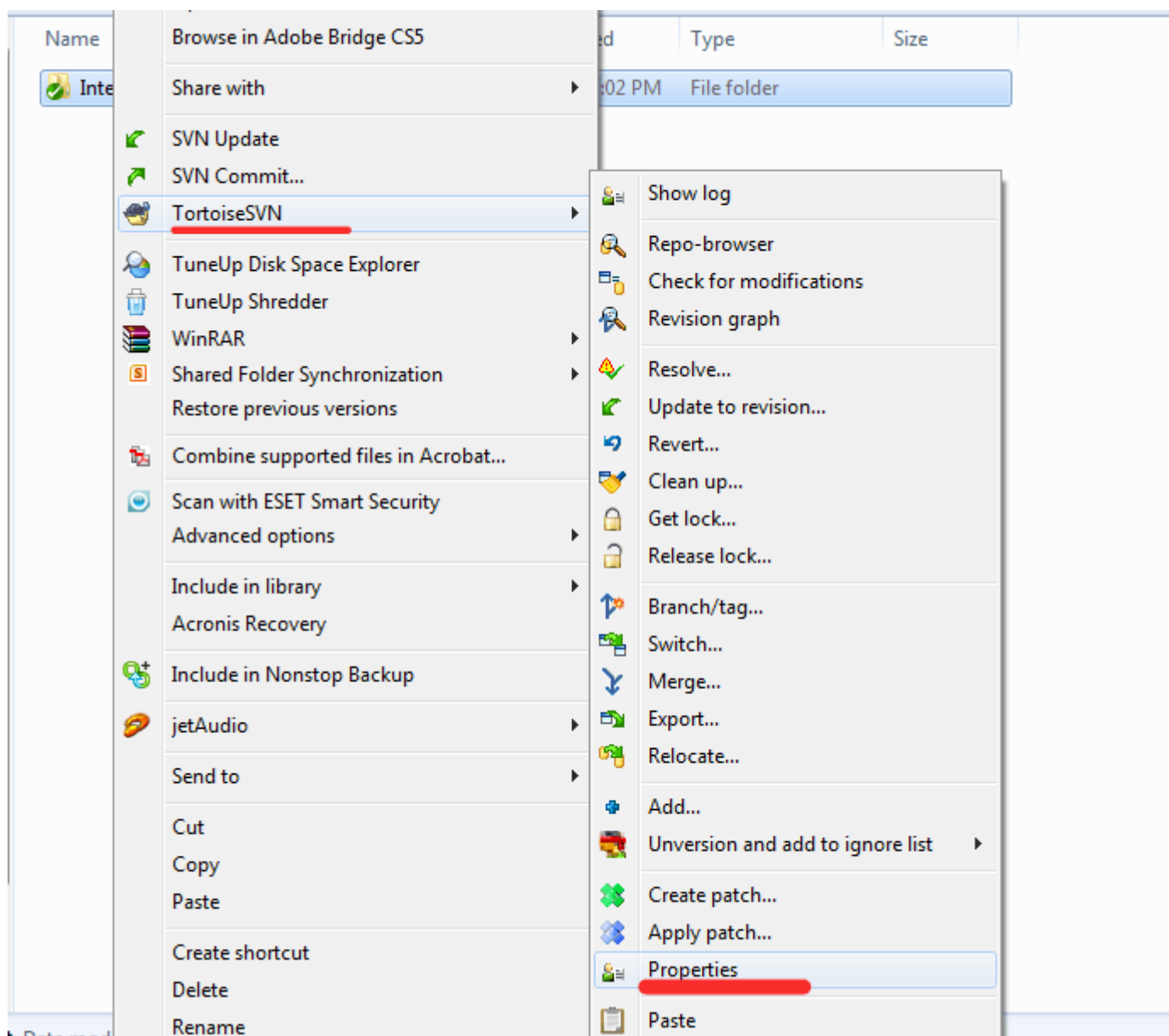
پیش نیاز اگر در مورد TortoiseSVN و سورس کنترل اطلاعات پایه ندارید، کتاب [مدیریت فایل‌های یک پروژه نرم افزاری با استفاده از Subversion](#) آقای نصیری را مطالعه کنید و همچنین سیستم پیگیری خطای [YouTrack](#) را نگاهی بیاندازید (البته اگر اطلاعی ندارید).

مقدمه

هنگام کار روی یک پروژه، باگ‌ها، وظیفه‌ها و موضوعاتی به شما واگذار می‌شود که باید آنها را انجام دهید. هنگام commit کردن تغییرات، برای مشخص شدن اینکه تغییرات مربوط به کدام Bug-Id بوده 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.



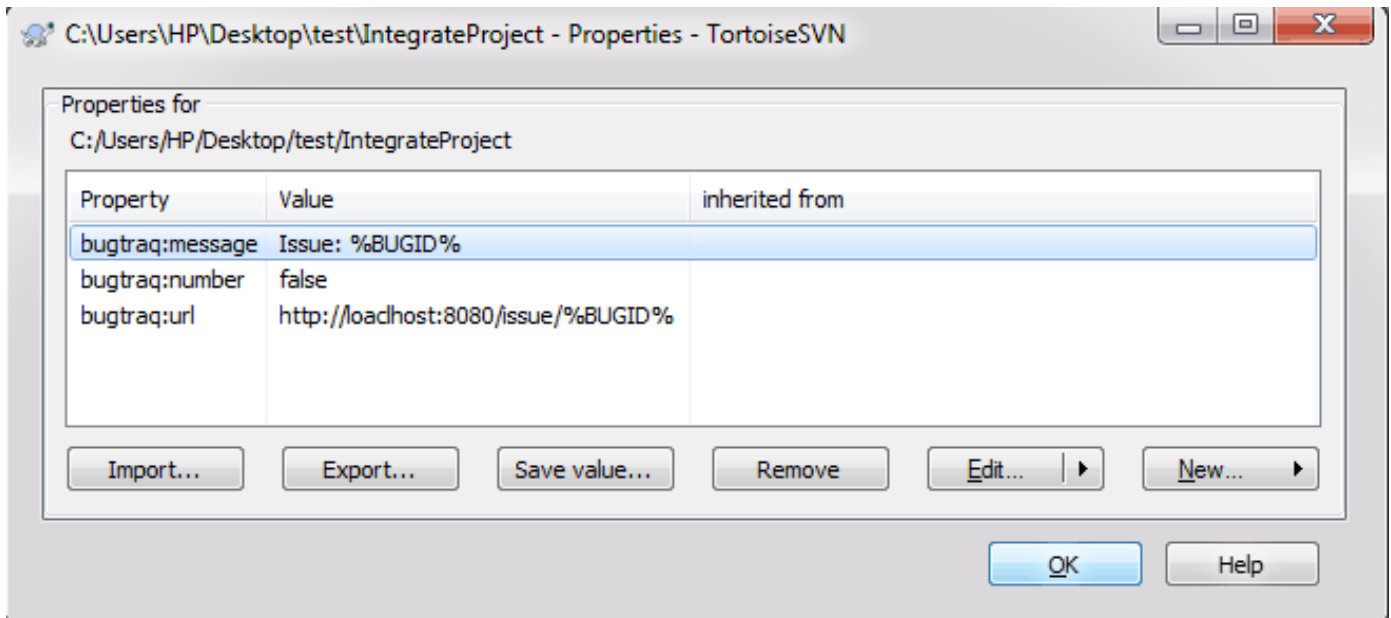
2- از پنجر باز شده دکمه New، گزینه Other را انتخاب کنید. در پنجره باز شده از منوی کشویی مربوط به Property Name، مقادیر

خصلت‌های زیر را تنظیم کنید: **bugtraq:url** : آدرس YouTrack Sever که به این صورت وارد می‌شود: %

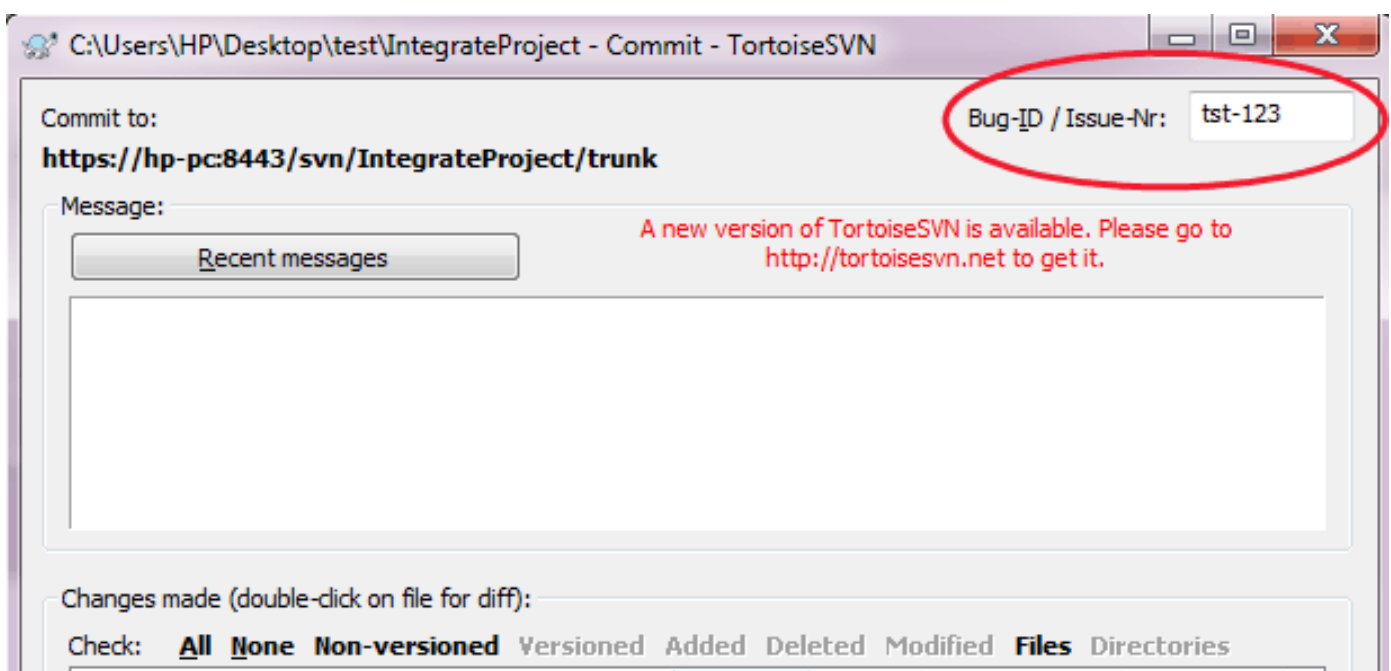
bugtraq:message : درو اقع الگویی پیامی هست که برای نگهداری Bug-Id استفاده می‌شود و باید شامل کلمه %BUGID% باشد.

مثلا: %BUGID% Issue:

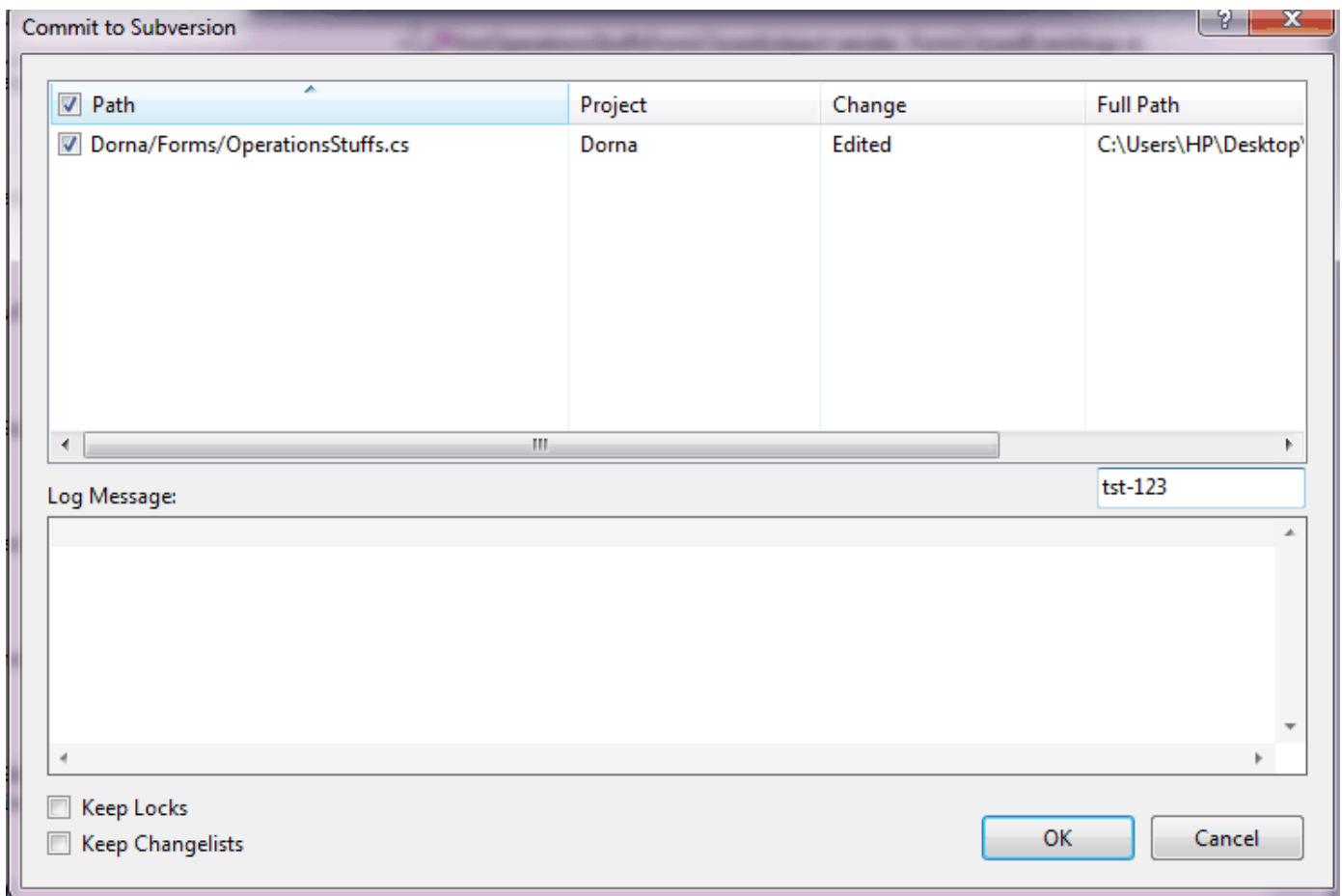
bugtraq:number : مقدار این خصلت را false وارد کنید؛ چون Bug-Id های YouTrack می‌توانند شامل عدد و حروف باشند.



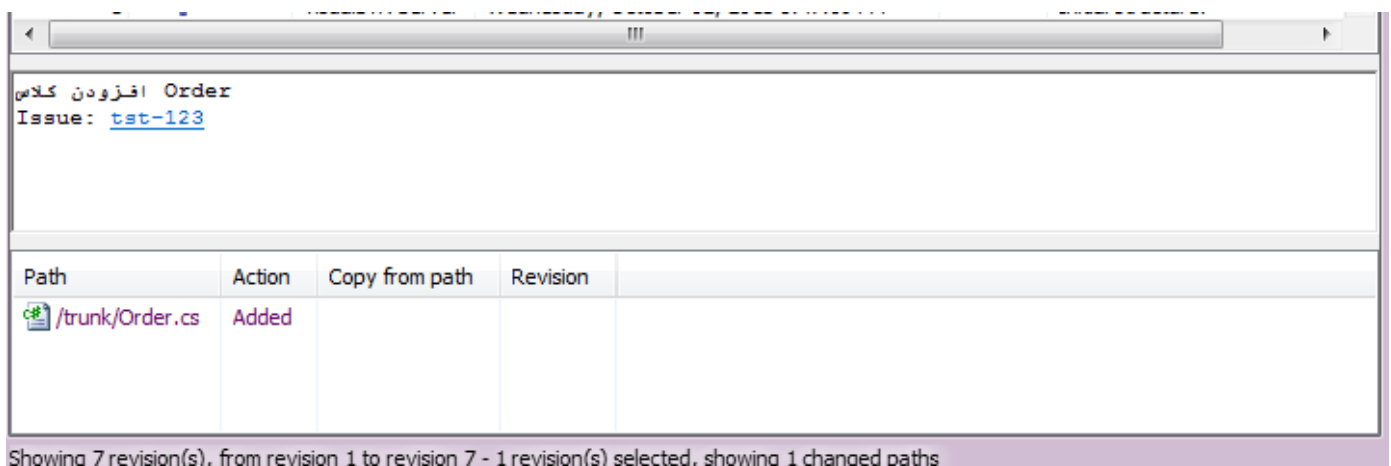
بعد از اینکه این سه خصلت را مقاردهی کردید، تغییرات را Commit کنید. همانطور که می‌بینید یک Textbox (بالا، سمت راست) اضافه شده که محل وارد کردن Bug-Id مربوط به تغییرات است. از این پس، می‌توانید Bug-Id یا Issue-Id های مربوط به هر تغییرات را در آن Textbox وارد کنید.



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



اکنون، در متن commit ها 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 بر روی 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 Bugtraq (Issue tracker integration) را انتخاب کنید.

Issue tracker
Specify the URL to access the issue tracker. Use %BUGID% as a placeholder for the real issue number.

URL:

☐ Remind me to enter a bug-ID

Message
Specify how the commit message should be built from the entered bug-ID. Use the placeholder %BUGID% for the real bug-ID. If you leave these settings empty, TortoiseSVN will use the regular expressions instead.

Message pattern:

Message label:

Bug-ID is: ☒ Arbitrary text ☐ Numeric

Insert message at: ☐ Top ☒ Bottom

Regular Expression
Enter the regular expression patterns for filtering out the bug-ID from a commit message.

Message part expression:

Bug-ID expression:

IBugTraqProvider

Provider uuid win32: uuid x64:

Provider parameters:

☐ Apply property recursively

OK Cancel Help

برای اطلاعات بیشتر در مورد این تنظیمات، داکيومنت [یکپارچه سازی با سیستم‌های Bug tracking / Issue Tracking](#) را مطالعه کنید.

در این مقاله با دو سیستم کنترل نسخه [git](#) و [SVN](#) آشنا شده و تفاوت های آن ها را برای تازه کاران بررسی می کنیم. ایده اولیه نوشتن این مقاله زمانی بود که برای یک پروژه ای، اعضای تیم ما دور هم جمع شده و در مورد ابزارهای مورد استفاده بحث کردند و یک عده از گیت و عده ای از SVN صحبت می کردند. بر این شدم که مقاله ای نوشته و ابتدا به معرفی آن ها و سپس به مزایا و معایب هر کدام بپردازیم.

امروزه، استفاده از سیستم های کنترل نسخه (Version Control System) رواج زیادی پیدا کرده است. این سیستم ها به شما اجازه می دهند تا تغییراتی را که در پروژه ایجاد می شوند، ضبط و ثبت کرده تا از تغییراتی که در سطح پروژه اتفاق می افتد 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 می دهند، 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 می توانید پروژه خود را به یک یا چند گام عقب تر برگردانید؟ آیا می توانید به هر یک از اعضای تیم دسترسی هایی را به قسمت هایی از پروژه تعیین کنید؟ می توانید تفاوت فایل های تغییر یافته را ببابید؟ 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](#) می تواند بسیار مفید باشد).

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

سیستم کنترل گیت: این سیستم در سال 2005 توسط لینوس توروالدز خالق لینوکس معرفی شد و از آن زمان تاکنون یکی از پر استفاده ترین سیستم های کنترل نسخه شناخته شده است. ویکی پدیا گیت را به این شکل تعریف می کند: « یک سیستم بازبینی توزیع شده با تاکید بر جامعیت داده ها، سرعت و پشتیبانی جهت توزیع کار. »

از معروف ترین سیستم های هاستینگ که از گیت استفاده می کنند، می توان به [گیت هاب](#) اشاره کرد.

اکثر سیستم های هاستینگ گیت، دو حالت را ارائه می دهند: عمومی: در این حالت کدهای شما به عموم بازدیدکنندگان نمایش داده می شود و دیگران هم می توانند 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 سیستم های مدیریتی این مورد را به صورت premium پشتیبانی می کنند. به این معنا که باید اجاره آن را به طور ماهانه پرداخت کنید. سیستم گیت هاب ماهی پنج دلار بابت آن دریافت می کند. سیستم دیگری که در این زمینه محبوبیت دارد سیستم [BitBucket](#) هست که اگر تیم شما کوچک است و در نهایت پنج نفر هستید، می توانید از حالت خصوصی به طور رایگان استفاده کنید ولی اگر اعضای تیم شما بیشتر شد، باید هزینه اجاره آن را که از 10 دلار آغاز می گردد، به طور ماهیانه پرداخت کنید.

پی نوشت: می توانید از سیستم های 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 می ماند.

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

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

Branching یا شاخه بندی: نویسنده ای مخزن اصلی می تواند با مفهومی با نام شاخه بندی کار کند. او با استفاده از این مفهوم، پروژه را به قسمت یا شاخه های مختلف تقسیم کرده و همچنین با ایجاد دسترسی های مختلف به کاربران اجازه تغییرات را بدهد. به عنوان مثال بخش های مختلف پروژه از قبیل بخش منطق برنامه، داده ها، رابط کاربری و ... می تواند باشد. بعد از انجام تغییرات روی یک شاخه می توانید درخواست merge شدن یا کل پروژه را داشته باشید. در عمل شاخه بندی، هیچ کدام از شاخه های 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 اصلاً 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 کنید.

نظرات خوانندگان

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

سلام؛ خیلی ممنون. چندتا سوال :
«پی نوشت: نه تنها در یک تیم بلکه بهتر هست در یک کار انفرادی هم از این سیستم ها استفاده کرد تا حداقل بازبینی روی پروژه های شخصی خود هم داشته باشیم.»
1- این یعنی اینکه اگر من بخوام برای خودم هم به تنهایی استفاده کنم و خصوصی هم باشه باید پول بدم؟ حالا اگر عمومی باشه می تونم به هیچ کس اجازه دسترسی ندم؟ فرق عمومی که اجازه دسترسی ندی با خصوصی تو چیه؟ دیدن و ندیدن کدها ؟
2-team foundation ماکروسافت هم برای اینکارهاست؟
3- می شه کمی بیشتر در این مورد توضیح بدید؟
«پی نوشت: میتوانید از سیستم های 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 می ماند.»

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

بحث git با هاست های عمومی git مثل github متفاوت هست. شما خودت هم می تونی 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: ۰۵/۱۴/۱۳۹۴ ۳۰:۱

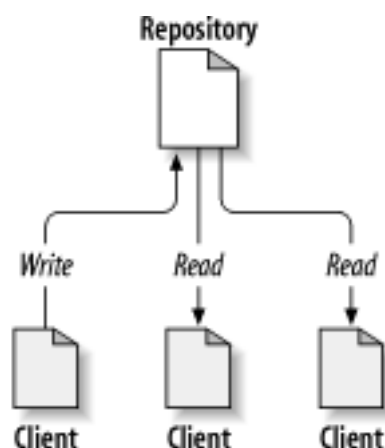
مبحث TFS کاملاً با مباحث سیستم های کنترل نسخه متفاوت است و یک سیستم ALM به حساب میاد نه VCS

فرقی نمی کند، پروژه عمومی همیشه نمایش داده می شود، این دسترسی ها 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 مجموعه ای از آیکن های کارت های اعتباری روبرو می شوید.

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.

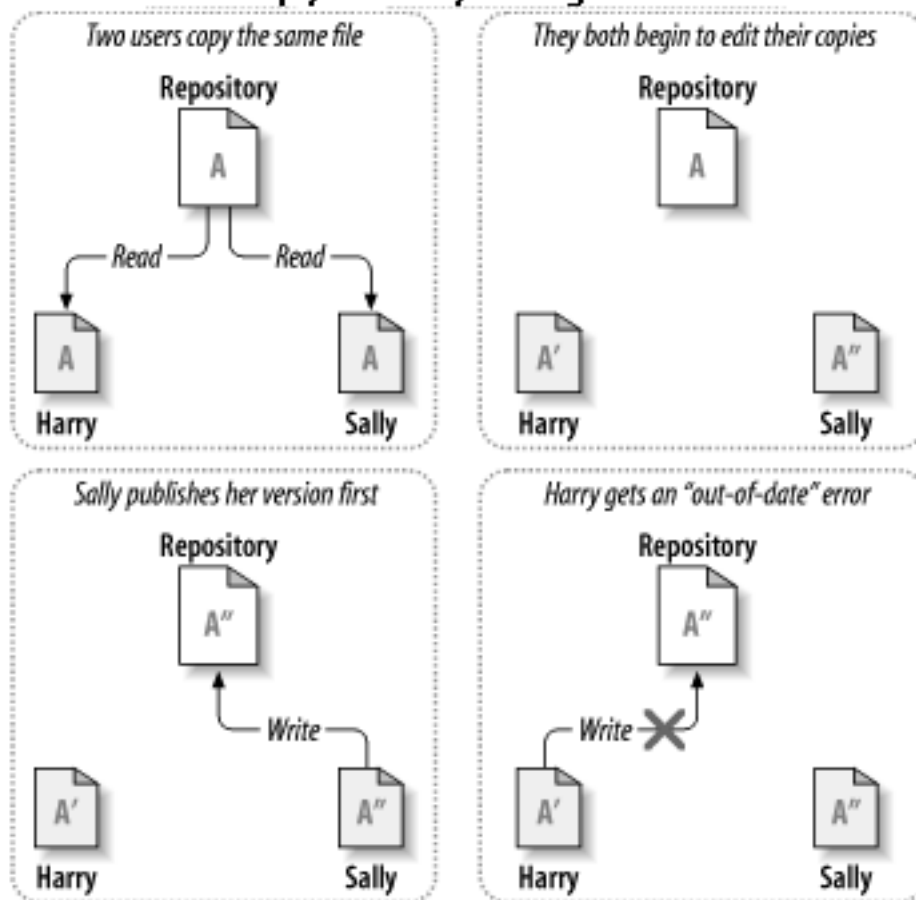
در قسمت قبلی ، اهمیت استفاده از سیستم های کنترل نسخه را بیان کردیم و مفاهیم پایه ای گیت را مورد بررسی قرار دادیم. در این قسمت مفاهیم پایه ای SVN را مورد بررسی قرار می دهیم. SVN مخفف عبارت SubVersion هست و یک سیستم کنترل نسخه ای رایگان و متن باز است که توسط شرکت کلاب نت حمایت می شود. به تعدادی از این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

در این سیستم فایل ها در یک مخزن Repository مرکزی ذخیره می شوند و با هر تغییری که در فایل ها و دایرکتوری ها ایجاد می شود، آن ها را ثبت می کند. این امکان به ما این اجازه را می دهد که نسخه ای قدیمی فایل ها را بازیابی کرده و تاریخچه ای اینکه فایل ها چگونه و چه موقع و توسط چه کسی تغییر کرده اند، به ما نشان دهد. تصویر سلسله مراتبی زیر به خوبی نحوه ارتباط کلاینت ها را به این مخزن نشان می دهد.



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

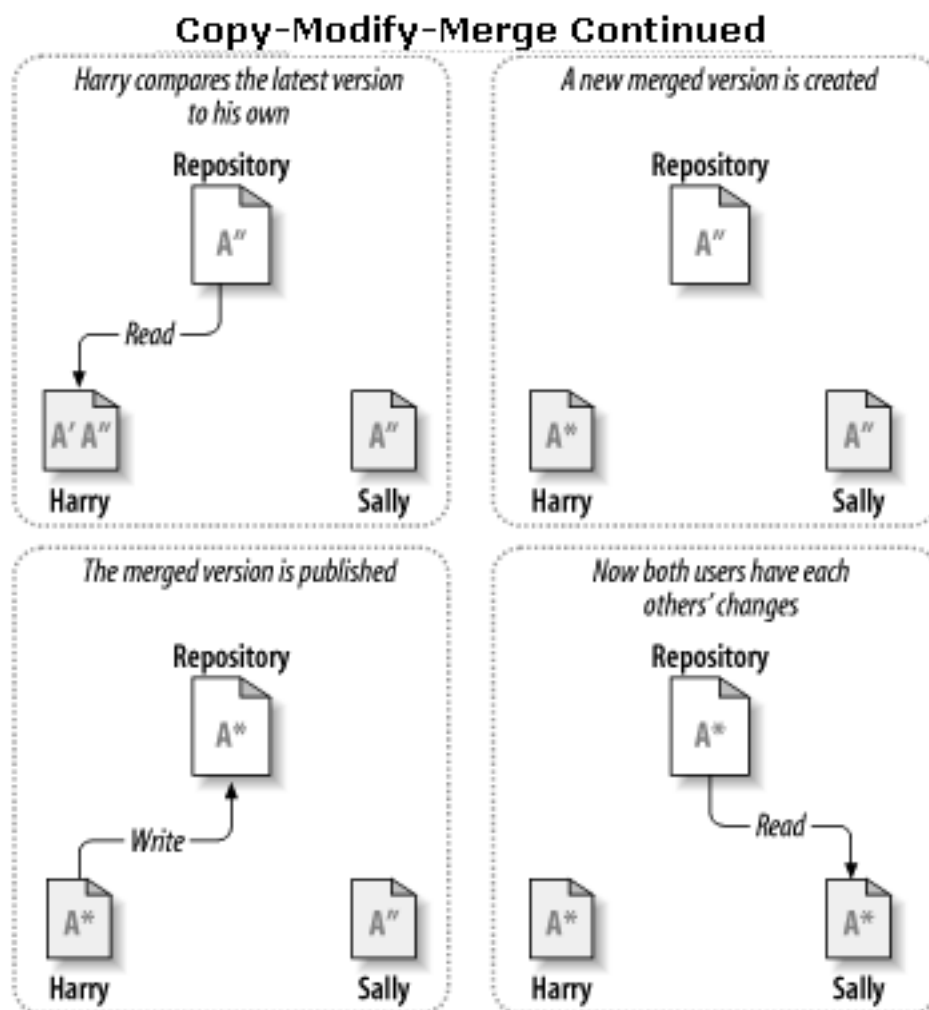
The Copy-Modify-Merge Solution



در تصویر بالا هری و سالی، یک کپی از مخزن موجود را گرفته و سپس هر کدام جداگانه بر روی کپی های خودشان مشغول به کار می شوند. سپس سالی کارش را زودتر به اتمام رسانده و مخزن را به روز می کند. بعد از آن، هری هم کارش به پایان می رسد و قصد به روز کردن مخزن را دارد ولی سیستم به او اجازه این کار را نمی دهد؛ 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 می تواند 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 فایل ها داده است همان تغییراتی است که سالی قبلا روی مخزن اصلی روی همان فایل ها اعمال کرده است، آیا در این حالت دریافت به روزرسانی های جدید باعث ایجاد تصادم می شود؟



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 نهایتاً به روشی کاملاً 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.

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 حرف های زیادی شنیده اید و احتمالا بیشتر آن ها در مورد گیت نظر مساعدتری داشته اند؛ ولی تفاوت هایی بین این دو سیستم هست که باید به نسبت هدف و نیازی که دارید آن را مشخص کنید. یکی از اصلی ترین این تفاوت ها این است که svn یک سیستم مرکزی است؛ ولی گیت اینگونه نیست که در ادامه تفاوت این دو مورد را تشریح می کنیم.

یک SVN یک مخزن مرکزی دارد که همه ی تغییراتی که روی کپی ها انجام می شود، 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 می شود، یک مخزن جداگانه به حساب می آید و Commit شدن تنها به مخزن کپی شده صورت می گیرد و در صورت pull request ادغام با مخزن اولیه خودش صورت می گیرد. دو. گیت به نسبت svn از پیچیدگی بیشتری برخوردار است؛ ولی برای پروژه های بزرگتر که کاربران زیادی با آن کار می کنند و احتمال شاخه بندی های زیادتر، در آن وجود دارد بهتر عمل می کند. موقعی که یک پروژه یا تیم کوچکی روی آن کار می کنند به دلیل commit شدن مستقیمی که svn دارد، کار راحت تر و آسان تر صورت می گیرد 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 می کند؛ سرعت بهتری نسبت به svn که یک کپی از زیر مجموعه ساختار اصلی ایجاد می کند دارد. چهار. شاخه بندی یک مفهوم اصلی و مهم در گیت به شمار می آید که اکثر کاربران همه روزه از آن استفاده می کنند و این اجازه را می دهد که تغییرات و تاریخچه فعالیت هر کاربر را بر روی هر شاخه، جداگانه ببینیم. در svn پیاده سازی شاخه ها یا تگ ها 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 می گردد. 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 می گیرد این است که به ازای هر فایل دو فایل موجود است یکی که همان فایل اصلی است که کاربر با آن کار می کند و دیگری یک فایل دیگر در شاخه svn. است که برای کمک به عملیاتی چون وضعیت، تفاوت ها، ثبت تغییرات به کار می رود. 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 می دهد. این عملیات شامل یافتن تفاوت ها، نمایش تاریخچه، ثبت تغییرات، ادغام شاخه ها و جابجایی بین شاخه ها می گردد. هفت. در سیستم SVN به دلیل ساختار درختی که دارد، می توانید زیر مجموعه ی یک مخزن را بررسی کنید ولی در سیستم گیت اینکار امکان پذیر نیست. البته باید به این نکته توجه داشت که برای یک پروژه ی بزرگ شما مجبور هستید همیشه کل مخزن را دانلود کنید. حتی اگر تنها نسخه ی خاصی از این زیرمجموعه را در نظر داشته باشید. به همین علت در شهرهایی که اینترنت گرانقیمت و یا سرعت پایین عرضه می شود، گیت به صرفه تر است و زمان کمتری برای دانلود آن می برد. **موارد تعریف شده زیر طبق گفته ویکی سایت Kernel.Org ذکر می شود:**

گیت از سیستم SVN سریعتر عمل می کند.
در سیستم گیت هر شاخه بندی کل تاریخچه خود را به دنبال دارد.
فایل git که تنظیمات مخزن داخلش قرار دارد، ساختار ساده ای دارد و به راحتی می توان در صورت ایجاد مشکل، آن را حل کرد و به ندرت هم پیش می آید که مشکلی برایش پیش بیاید.

پشتیبانی گیری از یک سیستم مرکزی مثل SVN راحت تر از پشتیبانی گیری از پوشه های توزیع شده در مخزن گیت است.
ابزارهای کاربری svn تا به الان پیشرفت های چشمگیری داشته است. پلاگین ها و برنامه های بیشتری نسبت به سیستم گیت دارد.
یکی از معروفترین این پلاگین ها، ابزار [tortoisesvn](http://tortoisesvn.net) است (البته ابزارهای گیت امروز رشد چشمگیرتری داشته اند که در قسمت اول نمونه های آن ذکر شد).

سیستم svn برای نسخه بندی و تشخیص تفاوت ها از یک سیستم ساده اعداد ترتیبی استفاده می کند که اولین ثبت با شماره یک آغاز شده و به ترتیب ادامه می یابد و برای کاربران هم خواندنش راحت است و هم قابل پیش بینی است. به همین جهت برای بررسی تاریخچه ها و دیگر گزارش ها تا حدی راحت عمل می کند. در سیستم شاخه بندی این سیستم شماره گذاری چندان مطلوب نیست و متوجه نمی شوید که این شاخه از کجا نشأت گرفته است. در حال حاضر برای پروژه ی موزیلا این عدد به 6 رقم رسیده است ولی در آن سمت، سیستم گیت از هش SH-1 استفاده می کند که یک رشته 40 کاراکتری است و 8 رقم اول آن به منشاء اشاره می کند که باعث می شود متوجه بشویم که این شاخه از کجا آمده است ولی از آنجا که این عدد یکتا ترتیبی نیست، برای خواندن و

گزارشگیری هایی که در SVN راحت صورت می گیرد، در گیت ممکن نیست یا مشکل است.
گیت رویدادهای ادغام و شاخه بندی را بهتر انجام می دهد.