

عنوان: نحوه ی دسترسی به یک سرور محلی TFS, از طریق اینترنت

نویسنده: میثم هوشمند

تاریخ: ۲۰:۵۵ ۱۳۹۲/۰۵/۰۵

آدرس: [www.dotnettips.info](http://www.dotnettips.info)

برچسب‌ها: TFS, Source control, Network

در محل کار برای مدیریت سورس پروژه‌هایی که در حال کار بر روی آن‌ها هستیم از TFS استفاده می‌کنیم. به دلیل اینکه عمده‌ی زمان کار ما بر روی پروژه‌ها محدود به وقتی هست که در شرکت حضور داریم، خیلی کم پیش آمده که نیاز به دسترسی به سرور خارج از شبکه‌ی داخلی به وجود بیاید.

اما در چند روز گذشته این نیاز به وجود آمده. خب اولین چیزی که به ذهن می‌رسد این هست که نیاز به یک Static IP و تعریف یک رکورد NAT در بخش تنظیمات مودم اینترنتی شبکه‌ی داخلی شرکت هست.

تا اینجا درست. اما Static IP بر روی سرویس ADLS شرکت تعریف نشده است و در هر بار اتصال به اینترنت IP جدید اما Valid به مودم تخصیص داده می‌شود. در یک شبکه‌ی Local می‌توانیم از طریق نام یک کامپیوتر به آن متصل بشویم. اما زمانی که خارج از آن شبکه قرار داشته باشیم انجام این کار مقدور نیست.

زمانی که در Visual Studio از منوی Team>Connect نام کامپیوتر مقصد - که همان سرور سورس کنترل باشد - را وارد می‌کنیم و از طریق کانکشنی که تعریف می‌کنیم به سورس کنترل متصل می‌شویم، این کانکشن بر اساس آدرس سرور Unique خواهد بود.

چنانچه خارج از شبکه‌ی Local بخواهیم از طریق Valid IP به همان سرور متصل بشویم، به دلیل اینکه Connection String جدید که بر اساس Valid IP می‌باشد با Connection String قبلی که بر اساس نام سرور می‌باشد متفاوت است، به همین دلیل ویژوال استودیو دو تعریف مجزا از این دو کانکشن خواهد داشت. بنابراین نمی‌توانیم پروژه‌ی مورد نظر خودمان را با کانکشن جدید، طبق روال گذشته مدیریت نماییم و عملیات Check-In و Check-Out ... را انجام دهیم.

برای رفع این مشکل می‌توانیم از طریق نگاشت نام سرور محلی به Valid IP اقدام نماییم. برای این کار، از مسیر C:\Windows\System32\drivers\etc\hosts فایل Hosts را به وسیله‌ی یک ویرایشگر متنی باز می‌کنیم و در انتهای خطوط موجود در فایل عبارت ذیل را وارد می‌نماییم.

VlidIP {TAB} Local\_Server\_Name

یعنی ابتدا آدرس آی پی سپس یک بار کلید Tab را فشار می‌دهیم و سپس نام کامپیوتر سرور محلی را درج می‌کنیم. بعد از این کار، در هر کجایی که نام سرور محلی را وارد نماییم، توسط Rule تعریف شده در فایل مذکور، نام سرور به آی پی مورد نظر نگاشت می‌شود.

\_ لازم به ذکر نیست که باید بر روی مودم اینترنتی شبکه‌ی داخلی مورد نظر باید توسط تعریف NAT درخواست هایی که روی پورت خاصی از مودم وارد می‌شوند را به همان شماره‌ی پورت بر روی رایانه‌ی سرور محلی منتقل کرد.

[ماخذ](#)

## نظرات خوانندگان

نویسنده: محسن خان  
تاریخ: ۲۲:۴۵ ۱۳۹۲/۰۵/۰۵

یکی از ملزومات دورکاری راه اندازی VPN هست. بعد با اتصال به اون از راه دور مثل این خواهد بود که شخص به شبکه داخلی متصل شده با همان تنظیمات و سطح دسترسی. VPN هم برای داخل کشور به داخل کشور مشکلی نداره (نه محدودیت سرعت و نه محدودیت دسترسی). تعدادی از شرکت های داخلی به همین نحو با کارمندان دورکار خودشون کار می کنند.

نویسنده: میثم هوشمند  
تاریخ: ۲۳:۳۵ ۱۳۹۲/۰۵/۰۵

دقیقا به نکته ی خوبی اشاره کردید. اما گاهی اوقات آدم یادش می رود!  
در حالتی که راه اندازی VPN سرور مقدور نباشد - به هر دلیلی - فکر میکنم که این راه هم خوب باشه.  
اما قطعاً VPN راه بهتری است.  
متشکرم

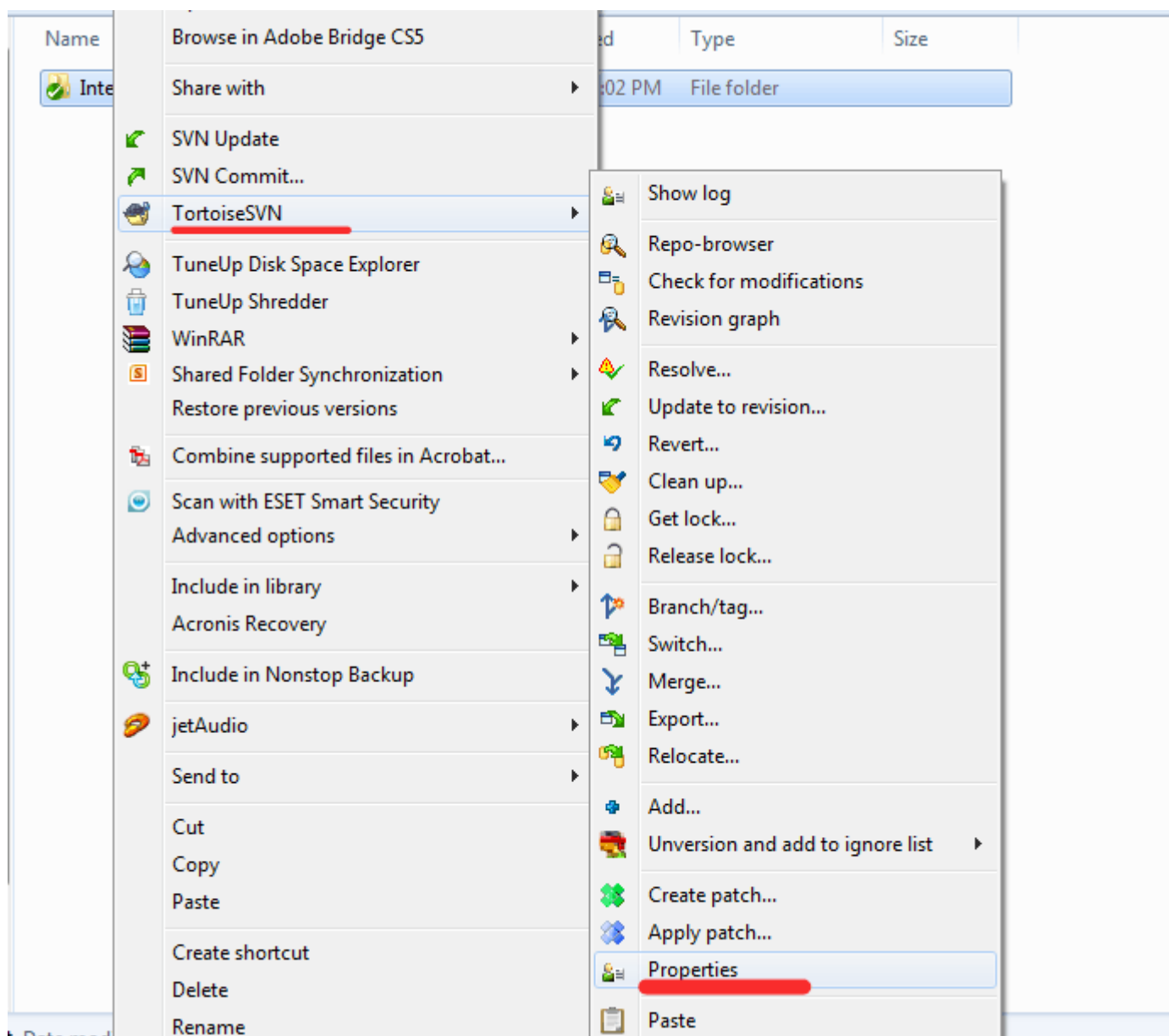
پیش نیاز اگر در مورد TortoiseSVN و سورس کنترل اطلاعات پایه ندارید، کتاب [مدیریت فایل‌های یک پروژه نرم افزاری با استفاده از Subversion](#) آقای نصیری را مطالعه کنید و همچنین سیستم پیگیری خطای [YouTrack](#) را نگاهی بیاندازید (البته اگر اطلاعی ندارید).

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

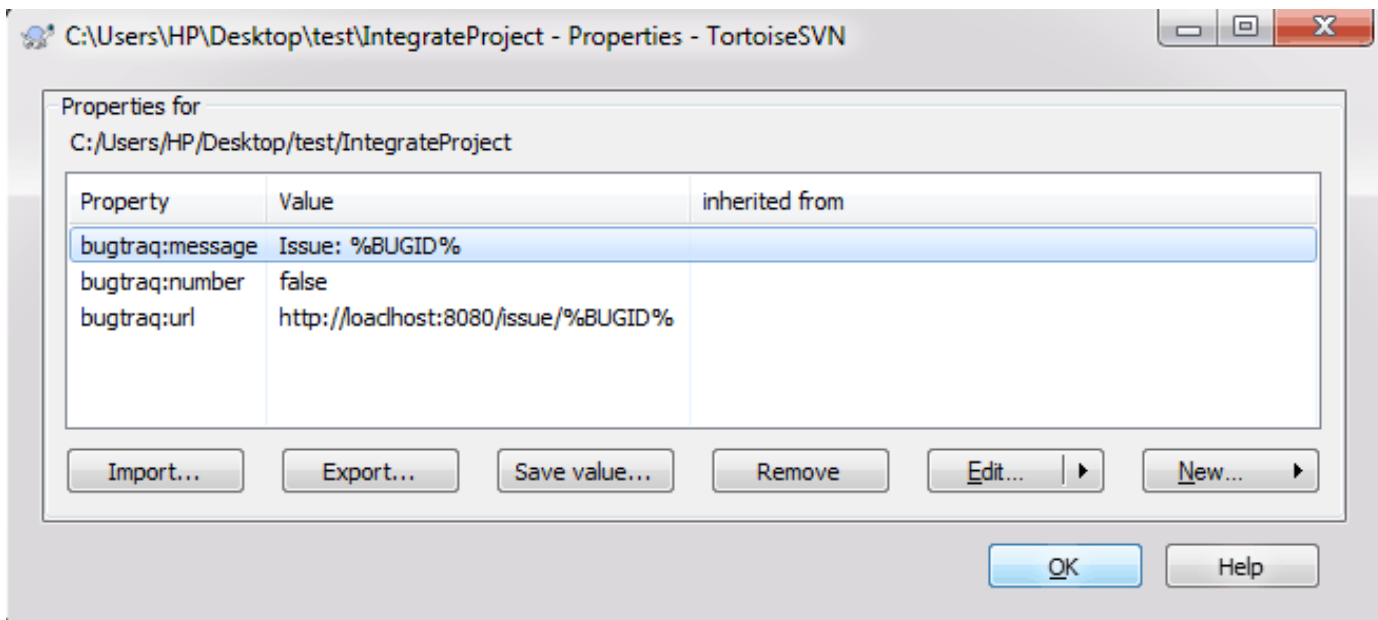
خصلت‌های زیر را تنظیم کنید: **bugtraq:url1** : آدرس YouTrack Sever که به این صورت وارد می‌شود: %

<http://localhost:8080/issue/%BUGID>

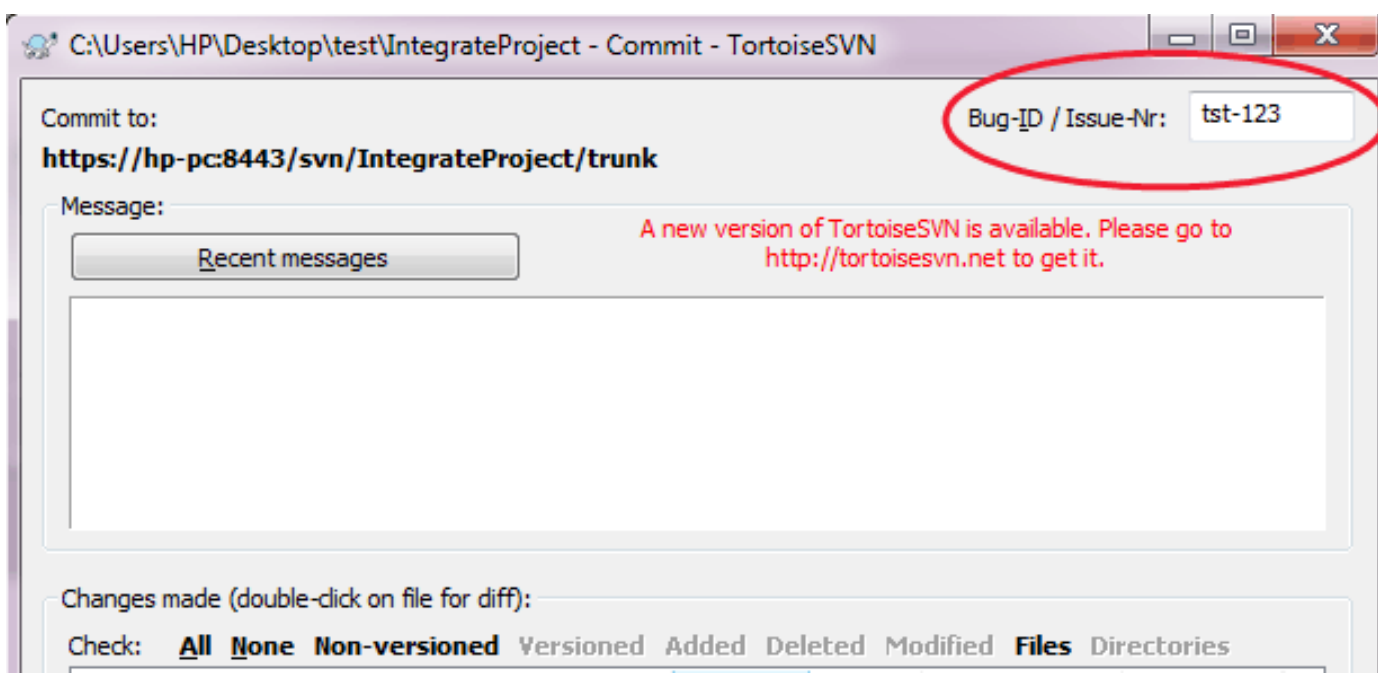
**bugtraq:message** : درو اقع الگویی پیامی هست که برای نگهداری Bug-Id استفاده می‌شود و باید شامل کلمه %BUGID% باشد.

مثلا: %BUGID%

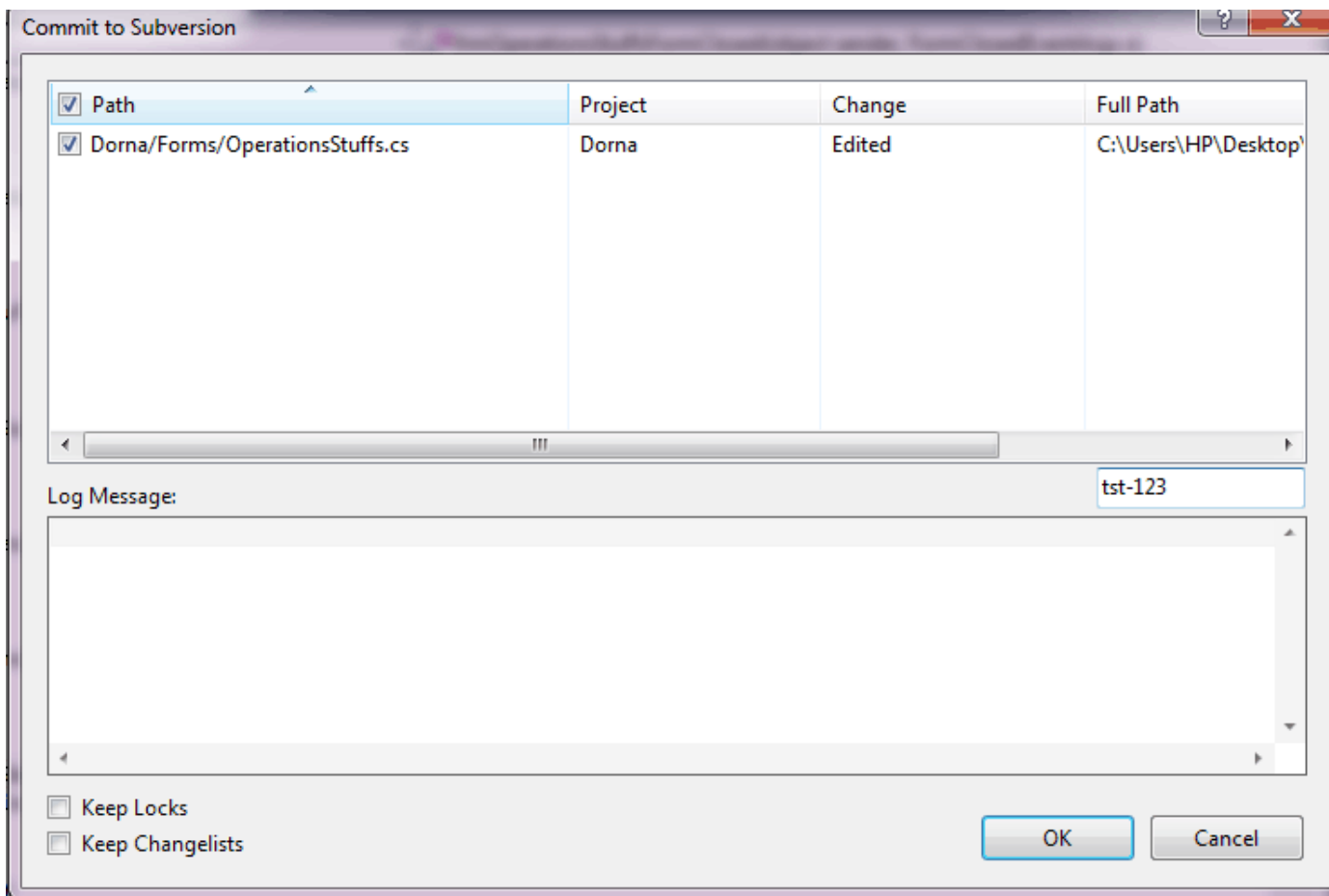
**bugtraq:number** : مقدار این خصلت را false وارد کنید؛ چون Bug-Id های YouTrack می‌توانند شامل عدد و حروف باشند.



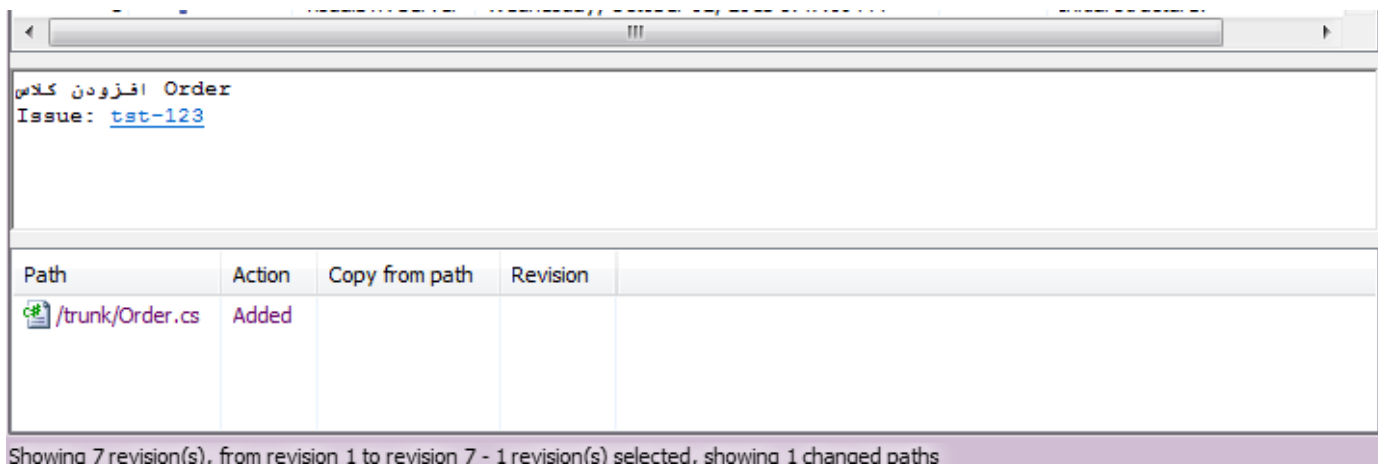
بعد از اینکه این سه خصلت را مقاردهی کردید، تغییرات را Commit کنید. همانطور که می‌بینید یک Textbox (بالا، سمت راست) اضافه شده که محل وارد کردن Bug-Id مربوط به تغییرات است. از این پس، می‌توانید Bug-Id یا Issue-Id های مربوط به هر تغییرات را در آن Textbox وارد کنید.



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.

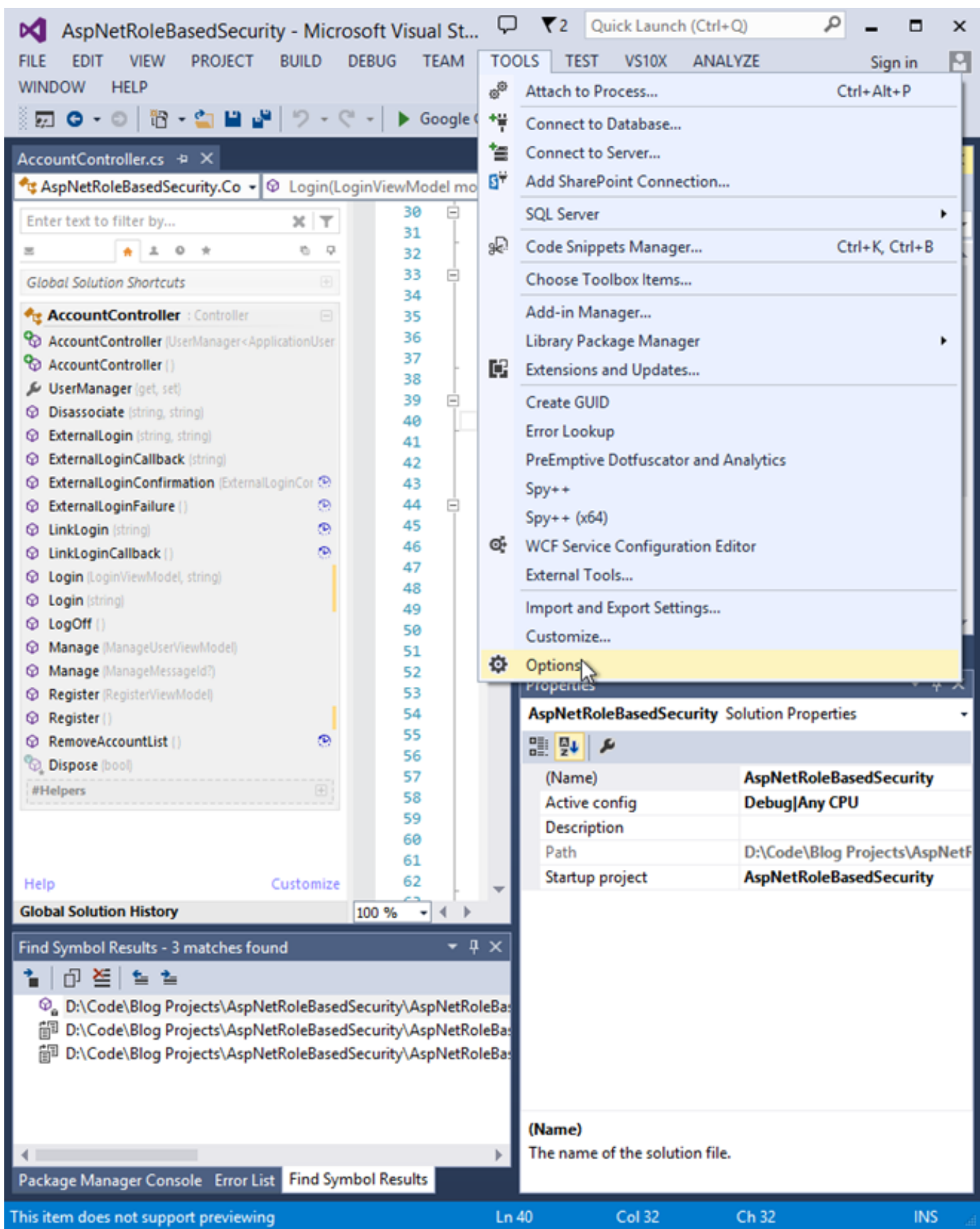
ابزار NuGet بسیار کار آمد و مفید است. یکی از مشکلات رایج هنگامی پیش می آید که پروژه را به همراه بسته های نصب شده به سورس کنترل push می کنید. با این کار حجم زیادی از فایل ها را به مخزن سورس کنترل آپلود می کنید و هنگام clone کردن پروژه توسط هر شخصی، این اطلاعات باید دریافت شوند. بدتر از این هنگامی است که برخی از بسته ها از سورس حذف می شوند و باید به اعضای تیم پروژه اطلاع دهید که چه بسته هایی باید دریافت و نصب شوند.

برای رفع این موارد به [NuGet Package Restore](#) وارد شوید.

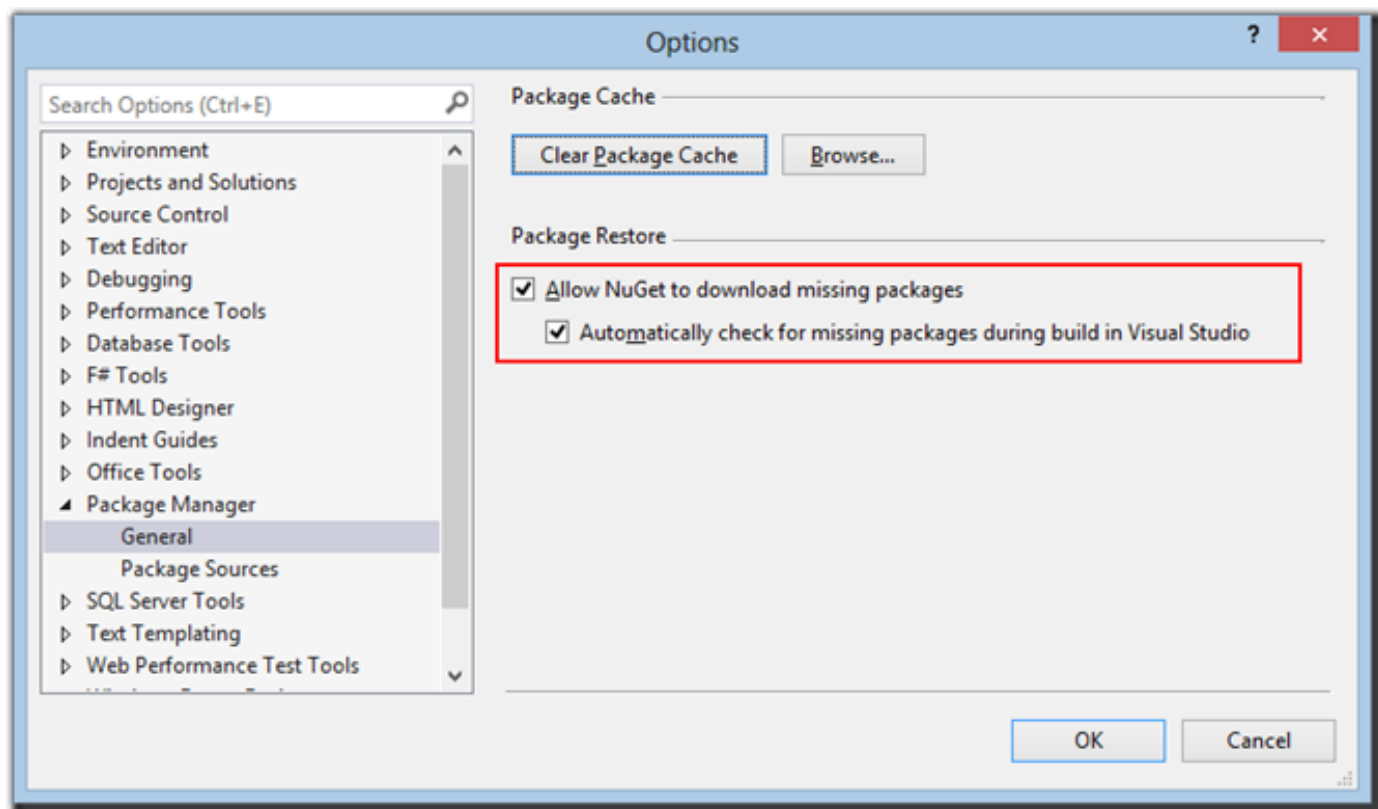
به ویژوال استودیو اجازه دهید بسته های NuGet را در صورت لزوم احیا کند

پیش از آنکه بتوانیم از قابلیت [Package Restore](#) استفاده کنیم باید آن را روی ماشین خود فعال کنیم. این کار روی هر ماشین باید انجام شود (per-machine requirement). بدین منظور به منوی Package Manager -> Options -> Tools بروید.





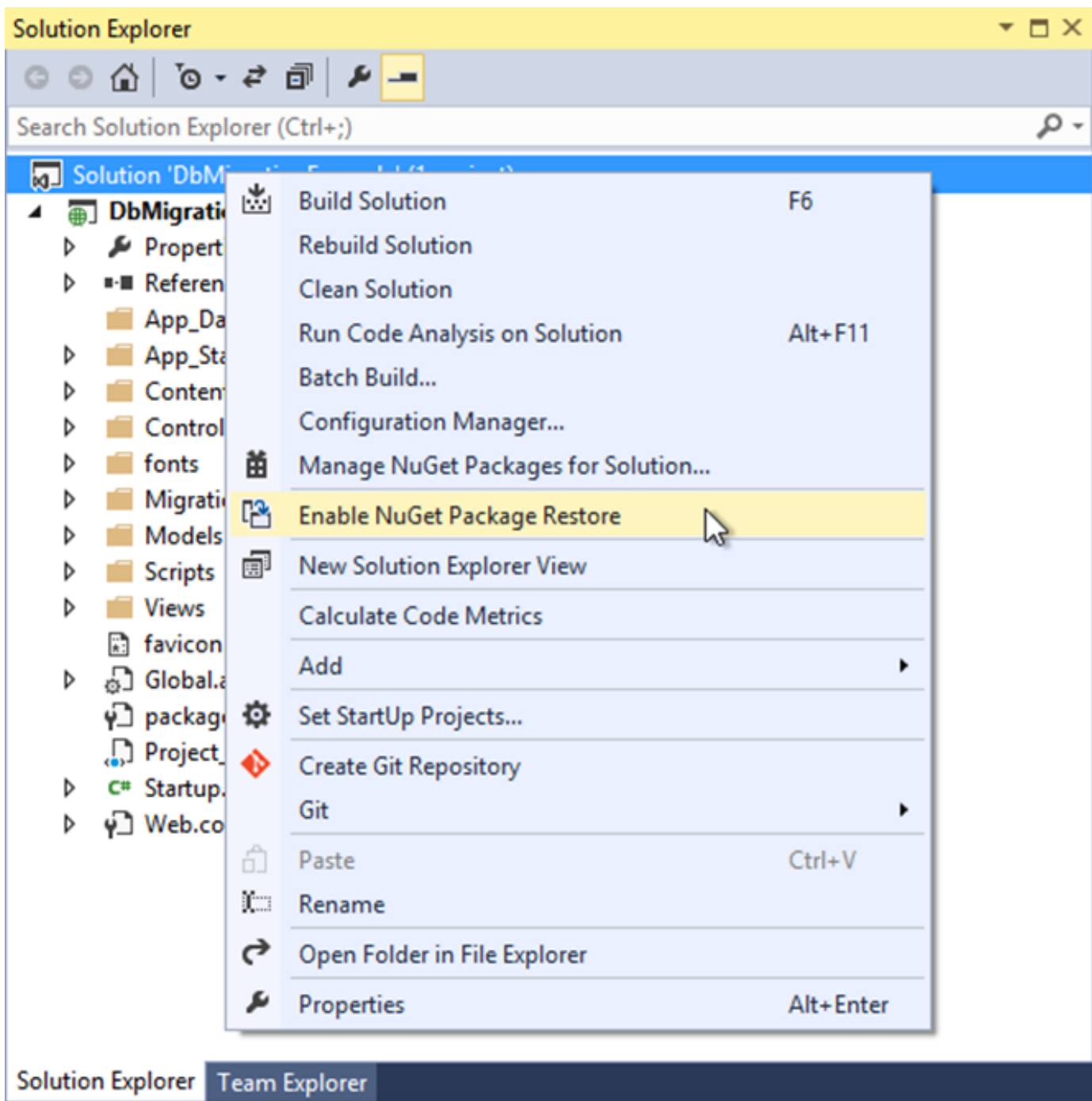
در دیالوگ باز شده تنظیمات مربوطه را مانند تصویر زیر بروز رسانی کنید.



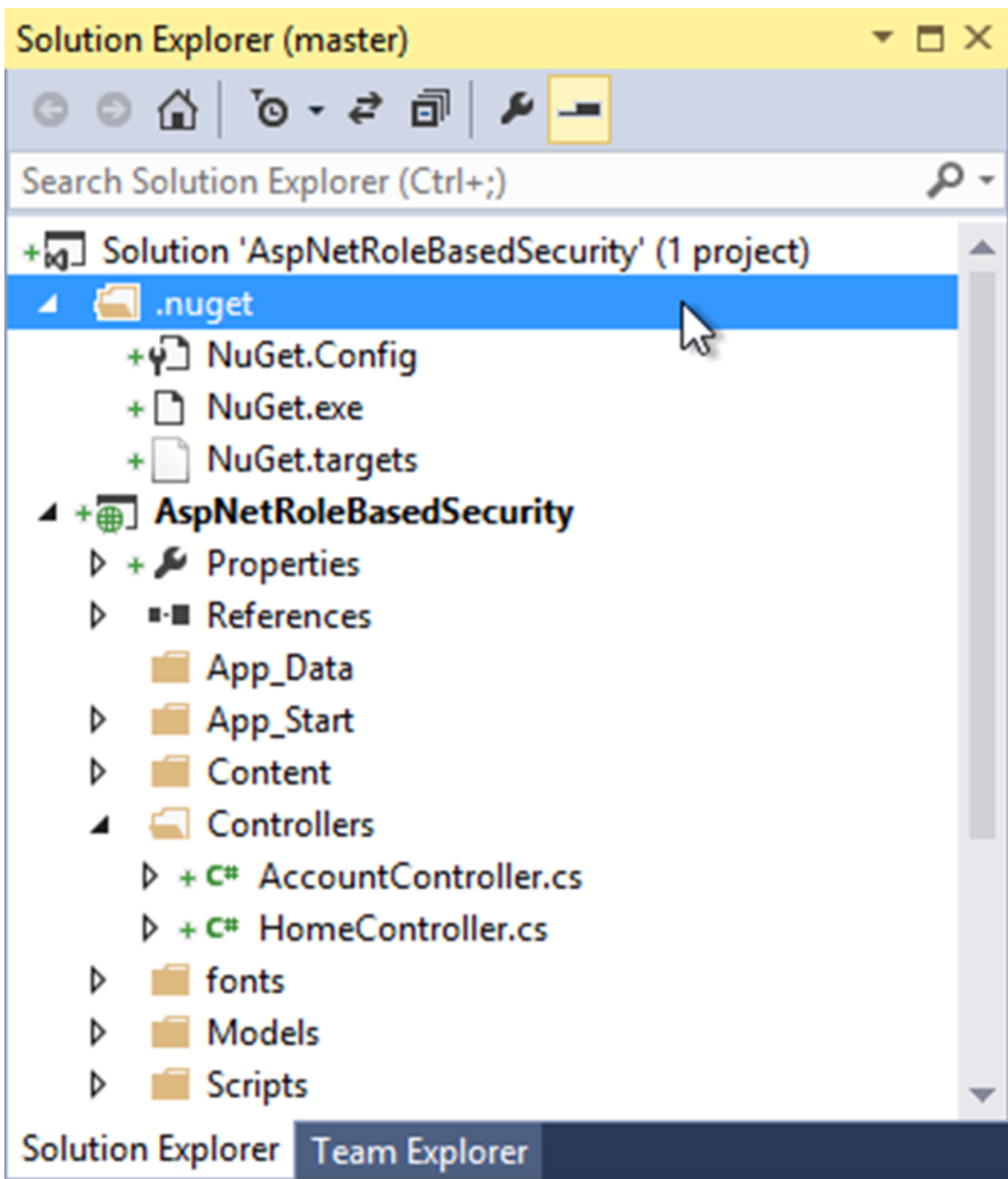
حال که ماشین ما برای بازیابی خودکار بسته‌های NuGet پیکربندی شده است، باید این قابلیت را برای Solution مورد نظر هم فعال کنیم.

#### فعال سازی NuGet Package Restore برای پروژه‌ها

بدین منظور روی Solution کلیک راست کنید و گزینه Enable Package Restore را انتخاب نمایید.



این کار ممکن است چند ثانیه زمان ببرد. پس از آنکه ویژوال استودیو پردازش های لازم را انجام داد، می توانید ببینید که پوشه جدیدی در مسیر ریشه پروژه ایجاد شده است.



همانطور که می بینید فایلی با نام NuGet.exe در این پوشه قرار دارد که باید به سورس کنترل آپلود شود. هنگامیکه شخصی پروژه شما را از سورس کنترل دریافت کند و بخواهد پروژه را Build کند، بسته های مورد نیاز توسط این ابزار بصورت خودکار دریافت و نصب خواهند شد.

مرحله بعد حذف کردن تمام بسته های NuGet از سورس کنترل است. برای اینکار باید فایل *gitignore* را ویرایش کنید. فرض بر

این است که سورس کنترل شما Git است، اما قواعد ذکر شده برای دیگر فریم ورک ها نیز صادق است. تنها کاری که باید انجام دهید این است که به سورس کنترل خود بگویید چه چیزهایی را در بر گیرد و از چه چیزهایی صرف نظر کند.

### ویرایش فایل `.gitignore` برای حذف بسته ها و شامل کردن `NuGet.exe`

یک پروژه معمولی ASP.NET MVC 5 که توسط قالب استاندارد VS 2013 ایجاد می شود شامل 161 فایل از بسته های مختلف می شود (در زمان تالیف این پست). این مقدار قابل توجهی است که حجم زیادی از اطلاعات غیر ضروری را به مخزن سورس کنترل اضافه می کند. با استفاده از نسخه پیش فرض فایل `.gitignore` (یا فایل های مشابه دیگر برای سورس کنترل های مختلف مثل TFS) تعداد فایل هایی که در کل به مخزن سورس کنترل ارسال می شوند بیش از 200 آیتم خواهد بود. قابل ذکر است که این تعداد فایل شامل فایل های اجرایی (binary) و متعلق به ویژوال استودیو نیست. به بیان دیگر نزدیک به 75% از فایل های یک پروژه معمولی ASP.NET MVC 5 که توسط VS 2013 ساخته می شود را بسته های NuGet تشکیل می دهد، که حالا می تواند بجای ارسال شدن به مخزن سورس کنترل، بصورت خودکار بازبایی و نصب شوند.

برای حذف این فایل ها از سورس کنترل، فایل `.gitignore` را ویرایش می کنیم. اگر از سورس کنترل های دیگری استفاده می کنید نام این فایل `.hgignore` یا `.tfignore` یا غیره خواهد بود. محتوای فایل شما ممکن است با لیست زیر متفاوت باشد اما جای نگرانی نیست. تنها تغییرات اندکی بوجود خواهیم آورد و مابقی محتویات فایل مهم نیستند.

### چشم پوشی از پوشه Packages

فایل `.gitignore` را باز کنید و برای نادیده گرفتن پوشه بسته های NuGet در سورس، خط زیر را به آن اضافه کنید.

```
packages*/
```

### استثنای برای در نظر گرفتن `NuGet.exe` ایجاد کنید

به احتمال زیاد فایل `.gitignore` شما از فایل هایی با فرمت `exe` چشم پوشی می کند. برای اینکه بسته های NuGet بتوانند بصورت خودکار دریافت شوند باید استثنای تعریف کنیم. فایل `.gitignore` خود را باز کنید و به دنبال خط زیر بگردید.

```
*.exe
```

سپس خط زیر را بعد از آن اضافه کنید. دقت داشته باشید که ترتیب قرارگیری این دستورات مهم است.

```
*.exe  
!NuGet.exe
```

دستورات بالا به Git می گوید که فایل های `exe` را نادیده بگیرد؛ اما برای فایل `NuGet.exe` استثناء قائل شود. انجام مرحله بالا انتخابی (optional) است. اگر کسی که پروژه را از مخزن سورس کنترل دریافت می کند قابلیت Package Restore را روی Solution فعال کند ابزار `NuGet.exe` دریافت می شود. اما با انجام این مراحل دیگر نیازی به این فعال سازی نخواهد بود، پس در کار اعضای تیم هم صرفه جویی کرده اید.

### اطلاع رسانی به اعضای تیم و مشتریان بالقوه

دیگر نیاز نیست بسته های NuGet را به مخزن سورس کنترل ارسال کنیم. اما باید به مخاطبین خود اطلاع دهید تا پیکربندی های لازم برای استفاده از قابلیت Package Restore را انجام دهند (مثلا در فایل `README.txt` پروژه).

## نظرات خوانندگان

نویسنده: Ara

تاریخ: ۲۲:۲۷ ۱۳۹۳/۰۲/۰۷

یک کار خوب داخل دیگه اینه که یک Local Package Source در شرکت داشته باشیم که دچار گیر کردن گاه به گاه ،مشکلات nuget تو ایران که بعضی وقتها گیر می‌کنه نیافتیم و package با سرعت بالا نصب بشوند

نویسنده: آرمین ضیاء

تاریخ: ۲۳:۲ ۱۳۹۳/۰۲/۰۷

میتونه رویکرد مناسبی باشه اما بهتر است که بسته‌های مورد نیاز از سرویس‌های معتبر مثل خود NuGet.org دریافت بشن تا انتشارات جدید در دسترس باشند. اگر منظورتون رو درست فهمیده باشم با این رویکرد یک کپی محلی از بسته‌ها خواهیم داشت. در صورتی که بسته‌ها نیاز به بروز رسانی داشته باشند نهایتا باز نیاز به دریافت پکیج‌ها از اینترنت است.

نویسنده: مسعود دانش پور

تاریخ: ۱۰:۱۱ ۱۳۹۳/۰۲/۰۸

به نظر بنده اگر به تایتل این نوشته مفید به "بیرون نگاه داشتن پکیج‌های NuGet از سورس کنترل Git" تغییر کنه بسیار عالی‌تر خواهد شد.

نویسنده: آرمین ضیاء

تاریخ: ۱۷:۴۹ ۱۳۹۳/۰۲/۰۸

با تشکر، عنوان بروز رسانی شد.

برخی از تنظیمات پروژه نباید به مخازن سورس کنترل ارسال شوند؛ حال یا نیازی به این کار نیست یا مقادیر تنظیمات محرمانه هستند. چند بار پیش آمده‌است که پروژه را از سورس کنترل دریافت و مجبور شده باشید رشته‌های اتصال و دیگر تنظیمات را مجدداً ویرایش کنید، چرا که توسعه دهندگان دیگری مثلاً فایل‌های Web/App.config خود را به اشتباه push کرده اند؟ حتی اگر تنظیمات پروژه محرمانه هم نباشند (مثلاً پسورد دیتابیس‌ها یا ایمیل‌ها) این موارد می‌توانند دردسر ساز شوند. بدتر از اینها هنگامی است که تنظیمات محرمانه را به مخازنی عمومی (مثلاً GitHub) ارسال می‌کنید!

یک فایل web.config معمولی را در نظر بگیرید (اطلاعات غیر ضروری حذف شده اند).

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  A bunch of ASP.NET MVC web config stuff goes here . . .
-->
<configuration>
  <connectionStrings>
    <add name="DefaultConnection" value="YourConnectionStringAndPassword"/>
  </connectionStrings>

  <appSettings file="PrivateSettings.config">
    <add key="owin:AppStartup"
value="AspNetIdentity2ExtendingApplicationUser.Startup,AspNetIdentity2ExtendingApplicationUser" />
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
    <add key="EMAIL_PASSWORD" value="YourEmailPassword"/>
  </appSettings>
</configuration>
```

در تنظیمات بالا یک رشته اتصال وجود دارد که ترجیحاً نمی‌خواهیم به سورس کنترل ارسال کنیم، و یا اینکه این رشته اتصال بین توسعه دهندگان مختلف متفاوت است. همچنین کلمه عبور یک ایمیل هم وجود دارد که نمی‌خواهیم به مخازن سورس کنترل ارسال شود، و مجدداً ممکن است مقدارش بین توسعه دهندگان متفاوت باشد. از طرفی بسیاری از تنظیمات این فایل متعلق به کل اپلیکیشن است، بنابراین صرف‌نظر کردن از کل فایل web.config در سورس کنترل گزینه جالبی نیست.

خوشبختانه کلاس ConfigurationManager راه حل هایی پیش پای ما می‌گذارد.

**استفاده از خاصیت configSource برای انتقال قسمت هایی از تنظیمات به فایل مجزا**

با استفاده از خاصیت configSource می‌توانیم قسمتی از تنظیمات (configuration section) را به فایل مجزا منتقل کنیم. بعنوان مثال، رشته‌های اتصال از مواردی هستند که می‌توانند بدین صورت تفکیک شوند.

بدین منظور می‌توانیم فایل تنظیمات جدیدی (مثلاً با نام *connectionStrings.config*) ایجاد کنیم و سپس با استفاده از خاصیت نام برده در فایل web.config به آن ارجاع دهیم. برای این کار فایل تنظیمات جدیدی ایجاد کنید و مقادیر زیر را به آن اضافه کنید (xml header یا هیچ چیز دیگری نباید در این فایل وجود داشته باشد، تنها مقادیر تنظیمات).

```
<connectionStrings>
  <add name="DefaultConnection" value="YourConnectionStringAndPassword"/>
</connectionStrings>
```



حال باید فایل web.config را ویرایش کنیم. رشته‌های اتصال را حذف کنید و با استفاده از خاصیت configSource تنها به فایل تنظیمات اشاره کنید.

```
<connectionStrings configSource="ConnectionStrings.config">
</connectionStrings>
```

دسترسی به رشته‌های اتصال مانند گذشته انجام می‌شود. به بیان دیگر تمام تنظیمات موجود (حال مستقیم یا ارجاع شده) همگی بصورت یکپارچه دریافت شده و به کد کلاینت تحویل می‌شوند.

```
var conn = ConfigurationManager.ConnectionStrings["DefaultConnection"];
string connString = conn.ConnectionString;
// etc.
```

در قطعه کد بالا، دسترسی به رشته‌های اتصال بر اساس نام، آبجکتی از نوع ConnectionStringSettings را بر می‌گرداند. خاصیت configSource برای هر قسمت از تنظیمات پیکربندی می‌تواند استفاده شود.

### استفاده از خاصیت file برای انتقال بخشی از تنظیمات به فایلی مجزا

ممکن است فایل تنظیمات شما (مثلا web.config) شامل مقادیری در قسمت <appSettings> باشد که برای کل پروژه تعریف شده اند (global) اما برخی از آنها محرمانه هستند و باید از سورس کنترل دور نگاه داشته شوند. در این سناریوها خاصیتی بنام file وجود دارد که مختص قسمت appSettings است و به ما اجازه می‌دهد مقادیر مورد نظر را به فایلی مجزا انتقال دهیم. هنگام دسترسی به مقادیر این قسمت تمام تنظیمات بصورت یکجا خوانده می‌شوند.

در مثال جاری یک کلمه عبور ایمیل داریم که می‌خواهیم محرمانه بماند. بدین منظور می‌توانیم فایل پیکربندی جدیدی مثلا با نام PrivateSettings.config ایجاد کنیم. این فایل هم نباید xml header یا اطلاعات دیگری داشته باشد، تنها مقادیر appSettings را در آن نگاشت کنید.

```
<appSettings>
  <add key="MAIL_PASSWORD" value="xspbqmurkjadeck"/>
</appSettings>
```

حال تنظیمات کلمه عبور را از فایل web.config حذف کنید و با استفاده از خاصیت file، به فایل جدید اشاره کنید.

```
<appSettings file="PrivateSettings.config">
  <add key="owin:AppStartup"
value="AspNetIdentity2ExtendingApplicationUser.Startup,AspNetIdentity2ExtendingApplicationUser" />
  <add key="webpages:Version" value="3.0.0.0" />
  <add key="webpages:Enabled" value="false" />
  <add key="ClientValidationEnabled" value="true" />
  <add key="UnobtrusiveJavaScriptEnabled" value="true" />
</appSettings>
```

دسترسی به تنظیمات appSettings مانند گذشته انجام می‌شود. همانطور که گفته شد ConfigurationManager بصورت خودکار اینگونه ارجاعات را تشخیص داده و تمام اطلاعات را بصورت یکجا در اختیار client code قرار می‌دهد.

```
var pwd = ConfigurationManager.AppSettings["MAIL_PASSWORD"];
```

### فایل‌های ویژه را به gitignore اضافه کنید

حال می‌توانیم فایل web.config را به سورس کنترل اضافه کنیم، فایل‌های ConnectionStrings.config و PrivateSettings.config را به فایل gitignore اضافه کنیم و پروژه را commit کنیم. در این صورت فایل‌های تنظیمات خصوصی به مخازن سورس کنترل ارسال نخواهند شد.

مستند سازی را فراموش نکنید!



مسئله اگر چنین رویکردی را در پیش بگیرید باید دیگران را از آن مطلع کنید (مثلا با افزودن توضیحاتی به فایل README.txt). بهتر است در فایل web.config خود هر جا که لازم است توضیحات XML خود را درج کنید و به توسعه دهندگان توضیح دهید که چه فایل هایی را روی نسخه های محلی خود باید ایجاد کنند و هر کدام از این فایل ها چه محتوایی باید داشته باشند.

## نظرات خوانندگان

نویسنده: رضایی  
تاریخ: ۱۳۹۳/۰۲/۰۸ ۰:۱۲

با سلام؛ ممنون بابت مطلب مفیدتون.  
میشه در خصوص gitignore توضیحاتی بفرمایید؟

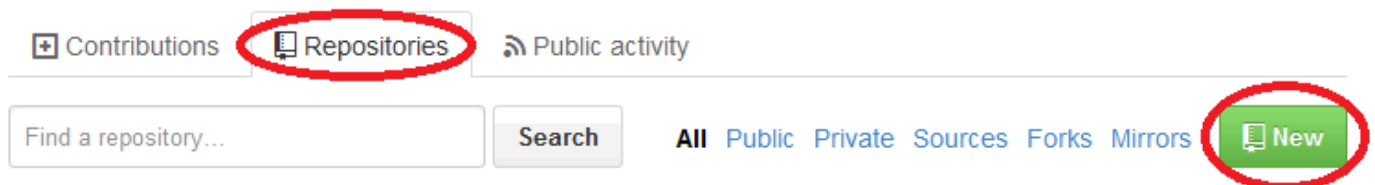
نویسنده: وحید نصیری  
تاریخ: ۱۳۹۳/۰۲/۰۸ ۰:۱۷

توضیحات بیشتر در سری Git » [آموزش سیستم مدیریت کد Git : استفاده به صورت محلی \(بخش دوم\)](#) «


از نگارش 2012 ویژوال استودیو، امکان کار با مخازن Git، به صورت یکپارچه و توکار و بدون نیاز به ابزارهای جانبی، توسط آن فراهم شده‌است. در ادامه قصد داریم به کمک این ویژگی توکار، نحوه‌ی ارسال یک پروژه‌ی از پیش موجود VS.NET را برای اولین بار به GitHub بررسی کنیم.

## تنظیمات مقدماتی GitHub

در ابتدا نیاز است یک مخزن کد خالی را در GitHub ایجاد کنید. برای این منظور به برگه‌ی Repositories در اکانت GitHub خود مراجعه کرده و بر روی دکمه‌ی New کلیک کنید:



سپس در صفحه‌ی بعدی، نام پروژه را به همراه توضیحاتی وارد نمائید و بر روی دکمه‌ی Create repository کلیک کنید. در اینجا سایر گزینه‌ها را انتخاب نکنید. نیازی به انتخاب گزینه‌ی READ ME و یا انتخاب مجوز و غیره نیست. تمام این کارها را در سمت پروژه‌ی اصلی می‌توان انجام داد و یا VS.NET فایل‌های ignore را به صورت خودکار ایجاد می‌کند. در اینجا صرفاً هدف، ایجاد یک مخزن کد خالی است.

**Owner**  **VahidN** / **Repository name**

Great repository names are short and memorable. Need inspiration? How about [turnt-octo-bugfixes](#).

**Description** (optional)

☒ **Public**  
Anyone can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.


☐ **Initialize this repository with a README**  
This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: **None** | Add a license: **None** ⓘ

**Create repository**

از اطلاعات صفحه‌ی بعدی، تنها به آدرس مخصوص GitHub آن نیاز داریم. از این آدرس در VS.NET برای ارسال اطلاعات به سرور استفاده خواهیم کرد:

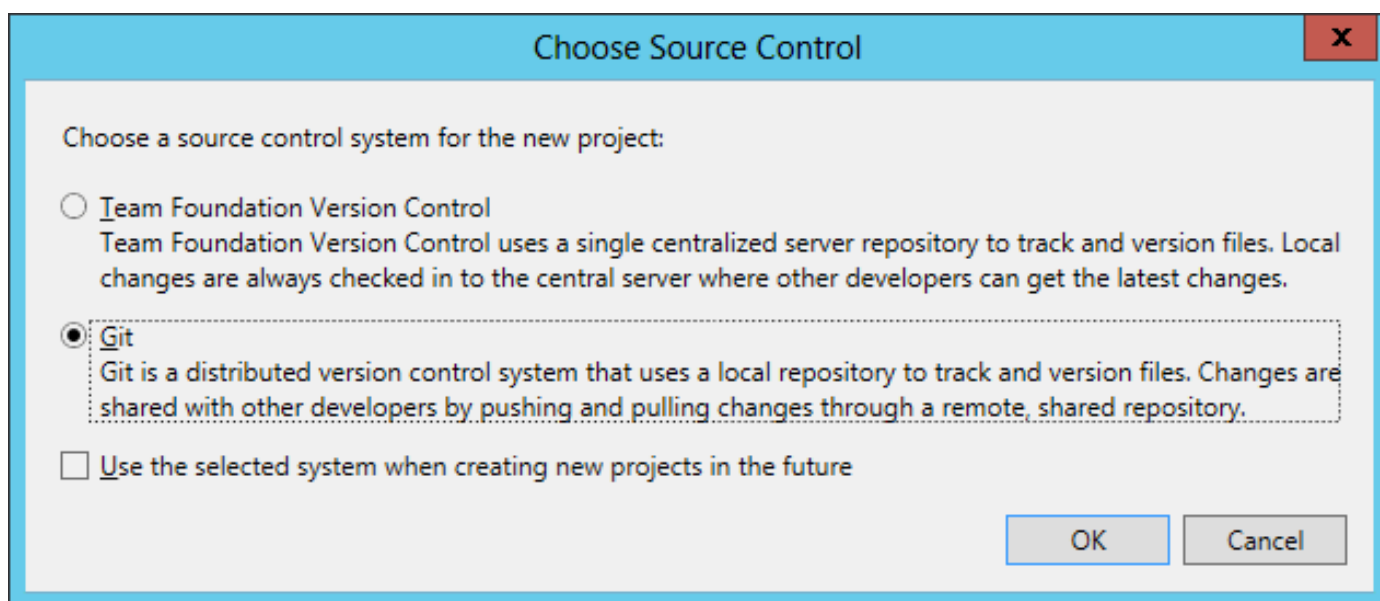
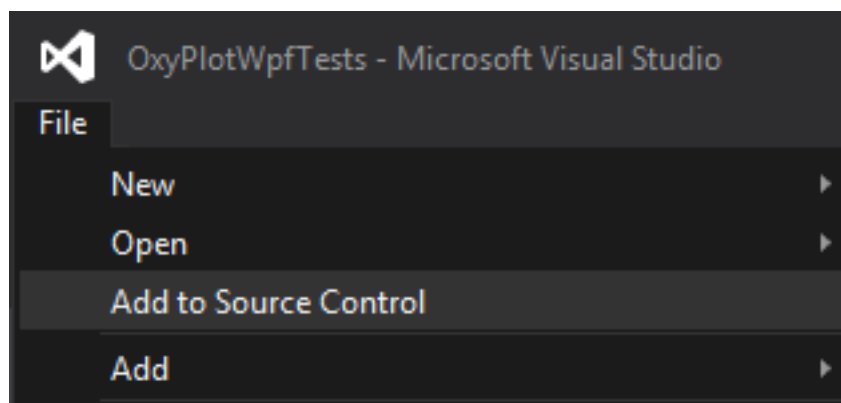
#### Quick setup — if you've done this kind of thing before

 **Set up in Desktop** or **HTTPS** **SSH**

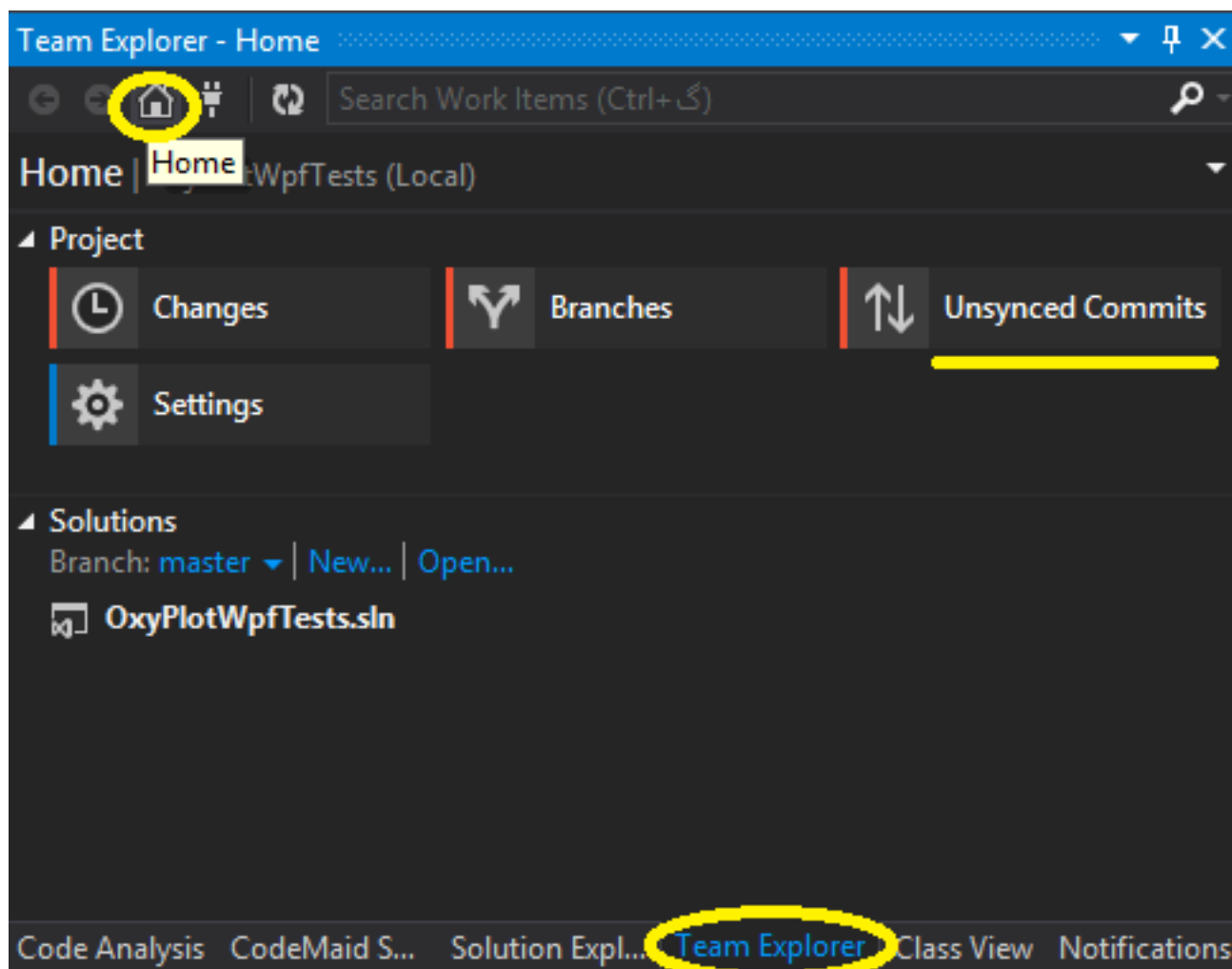
We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

#### تنظیمات VS.NET برای ارسال پروژه به مخزن GitHub

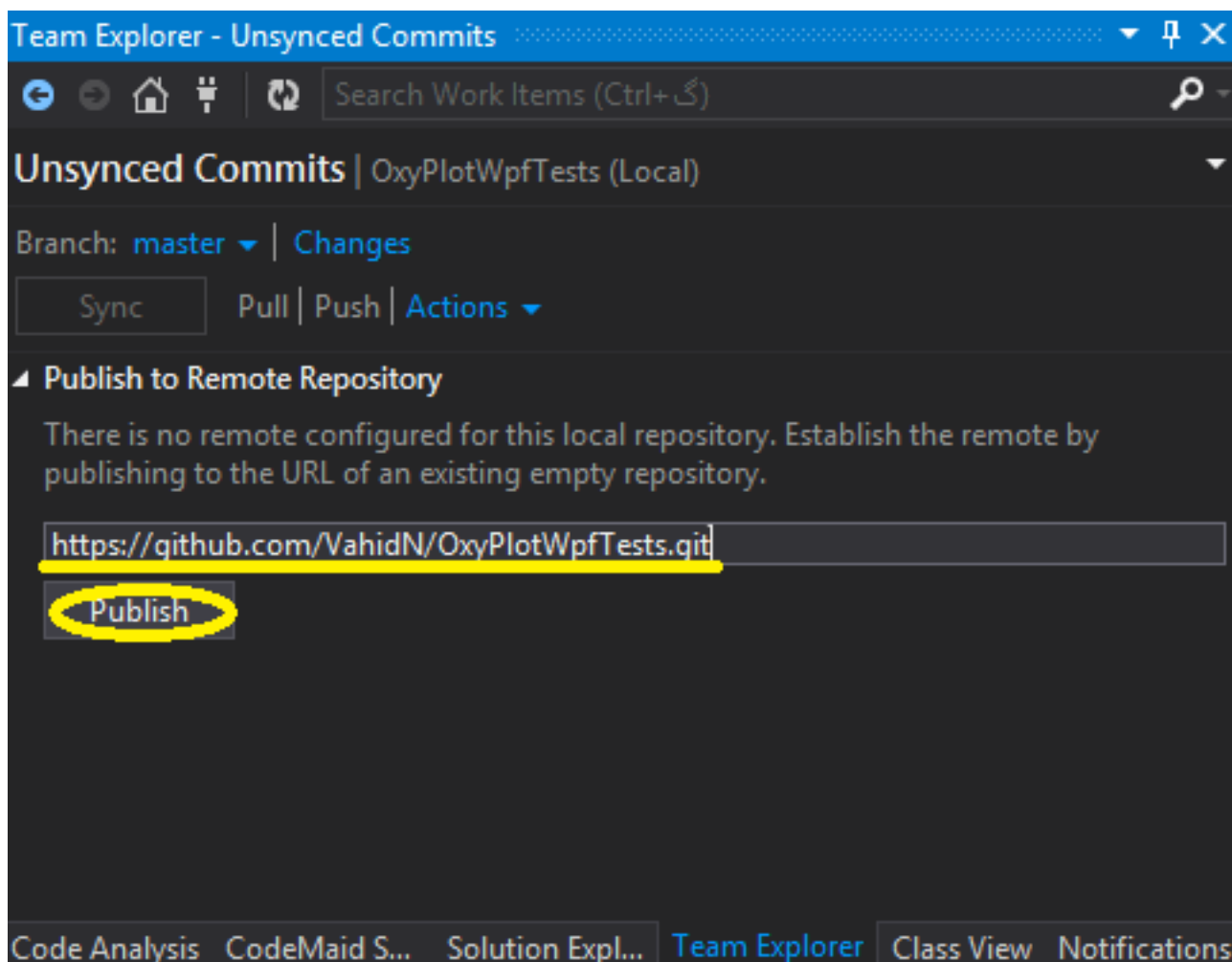
پس از ایجاد یک مخزن کد خالی در GitHub، اکنون می‌توانیم پروژه‌ی خود را به آن ارسال کنیم. برای این منظور از منوی File، گزینه‌ی Add to source control را انتخاب کنید و در صفحه‌ی باز شده، گزینه‌ی Git را انتخاب نمایید:



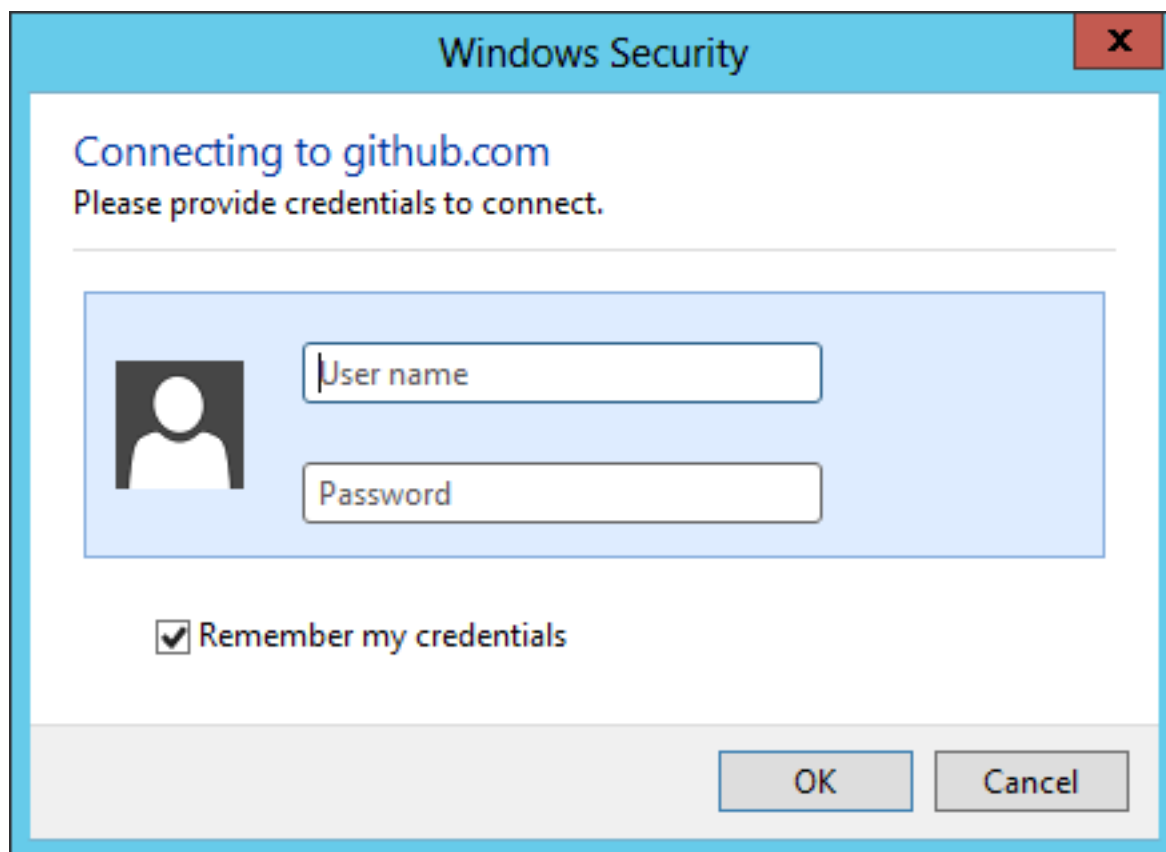
سپس در کنار برگه‌ی Solution Explorer، برگه‌ی Team Explorer را انتخاب کنید. در اینجا بر روی دکمه‌ی Home در نوار ابزار آن کلیک کرده و سپس بر روی دکمه‌ی Unsynced commits کلیک نمایید.



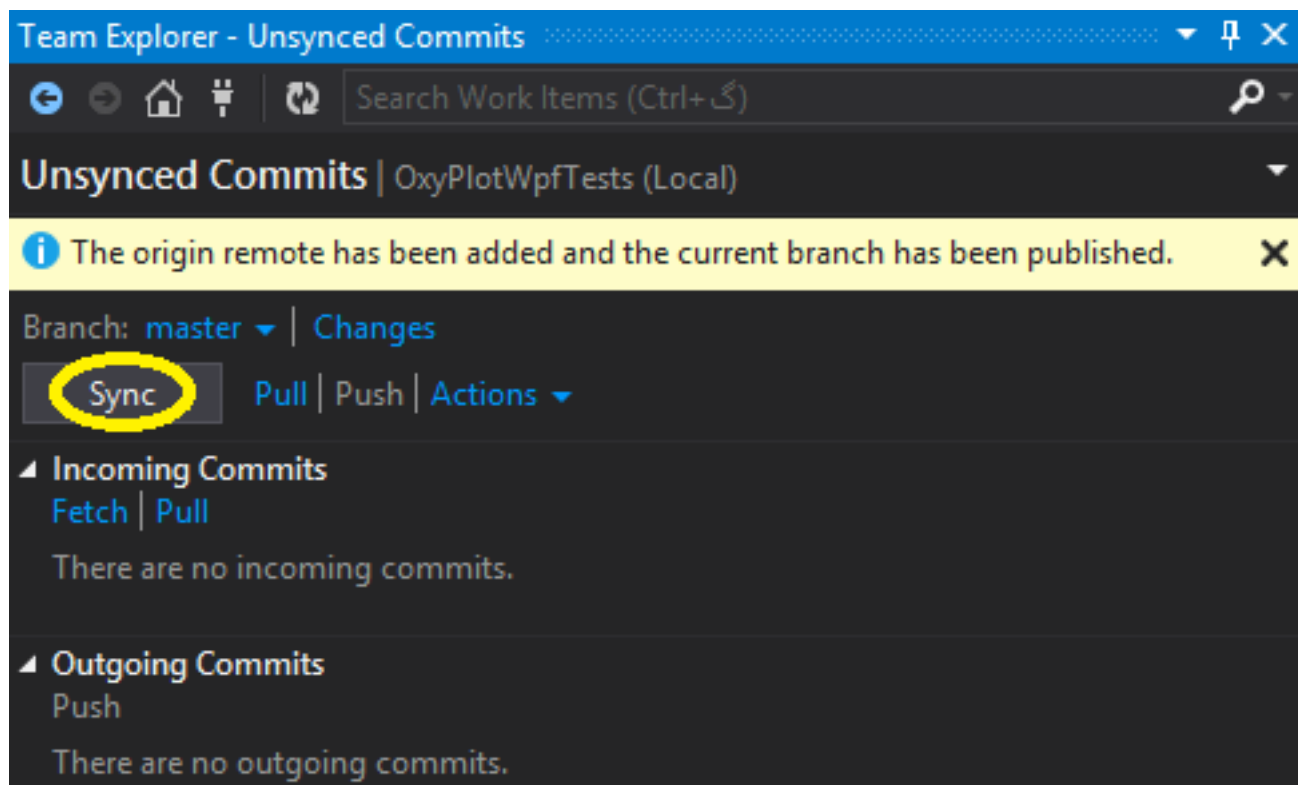
در ادامه در صفحه‌ی باز شده، همان آدرس مخصوص مخزن کد جدید را در GitHub وارد کرده و بر روی دکمه‌ی Publish کلیک کنید:



در اینجا بلافاصله صفحه‌ی لاگینی ظاهر می‌شود که باید در آن مشخصات اکانت GitHub خود را وارد نمایید:

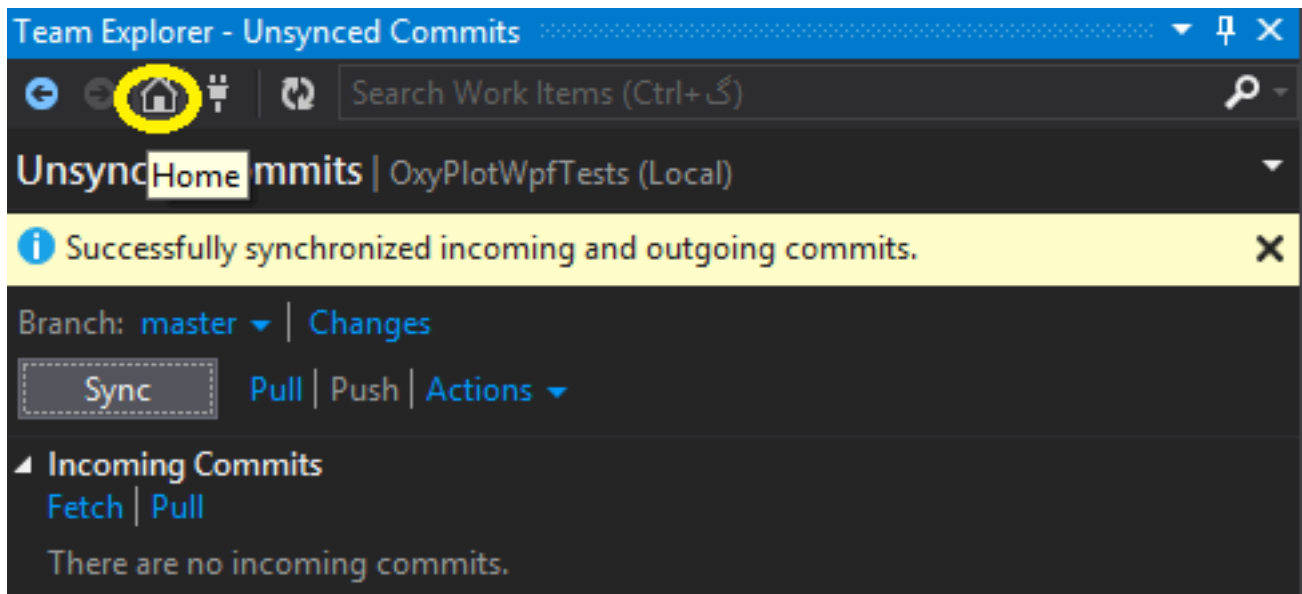


به این ترتیب عملیات Publish اولیه انجام شده و تصویر ذیل نمایان خواهد شد:

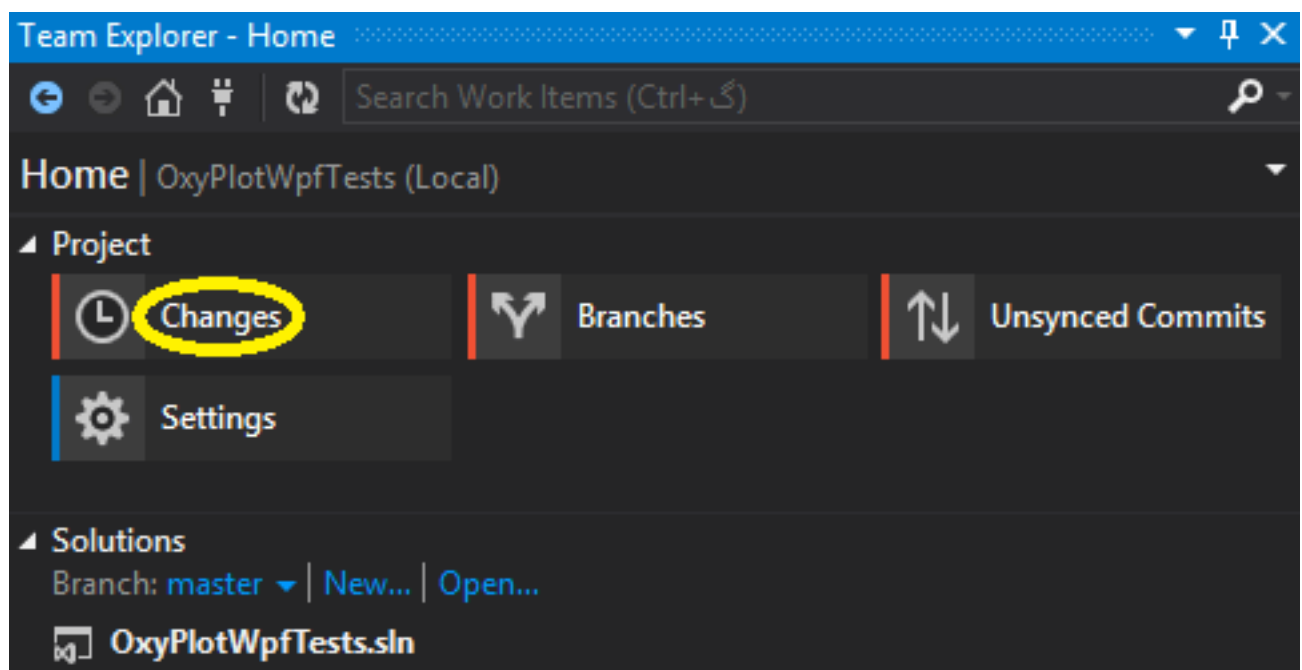




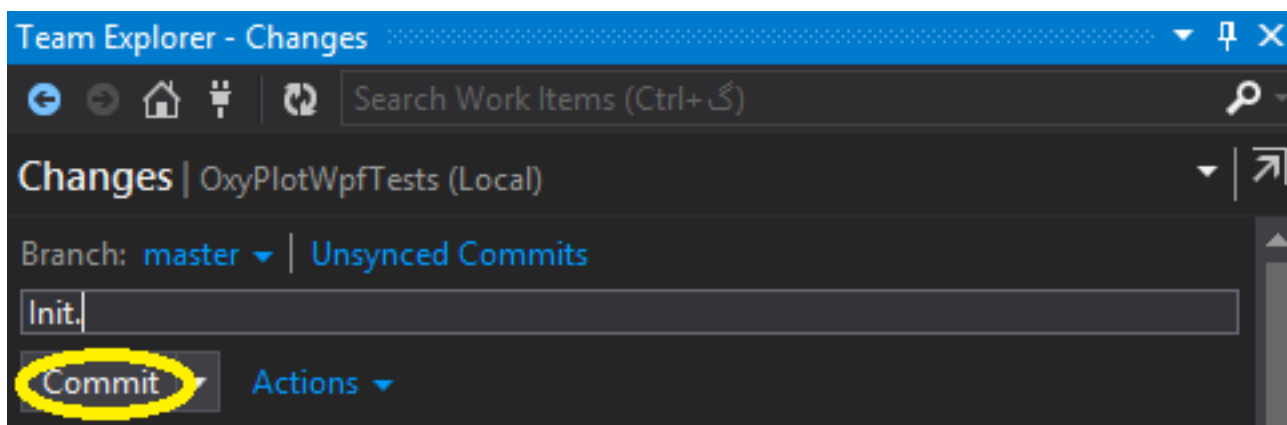
در اینجا بر روی دکمه‌ی Sync کلیک کنید. به این ترتیب مخزن کد GitHub به پروژه‌ی جاری متصل خواهد شد:



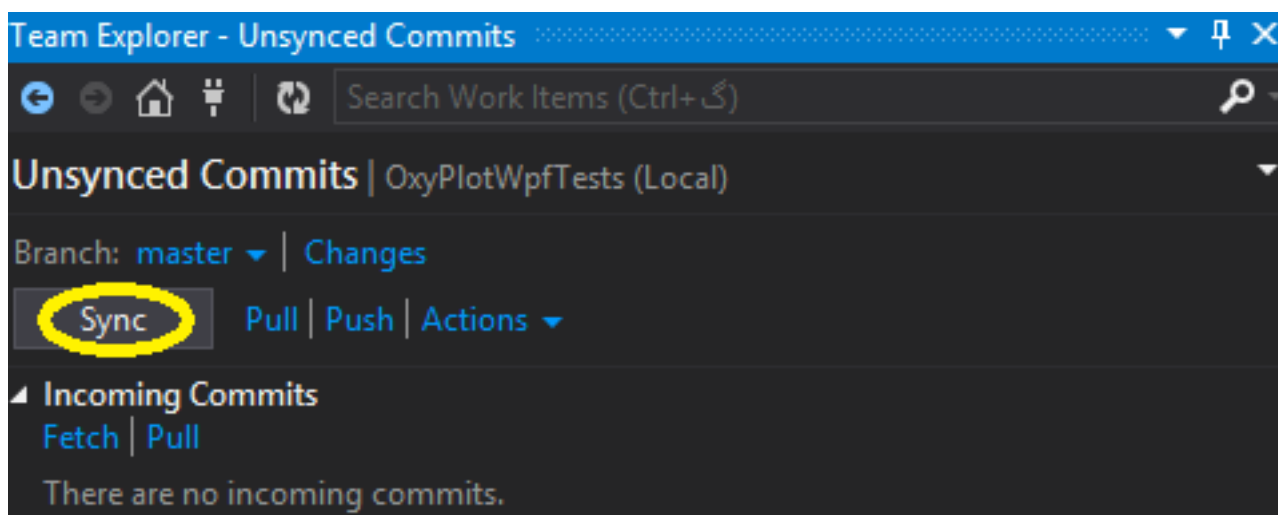
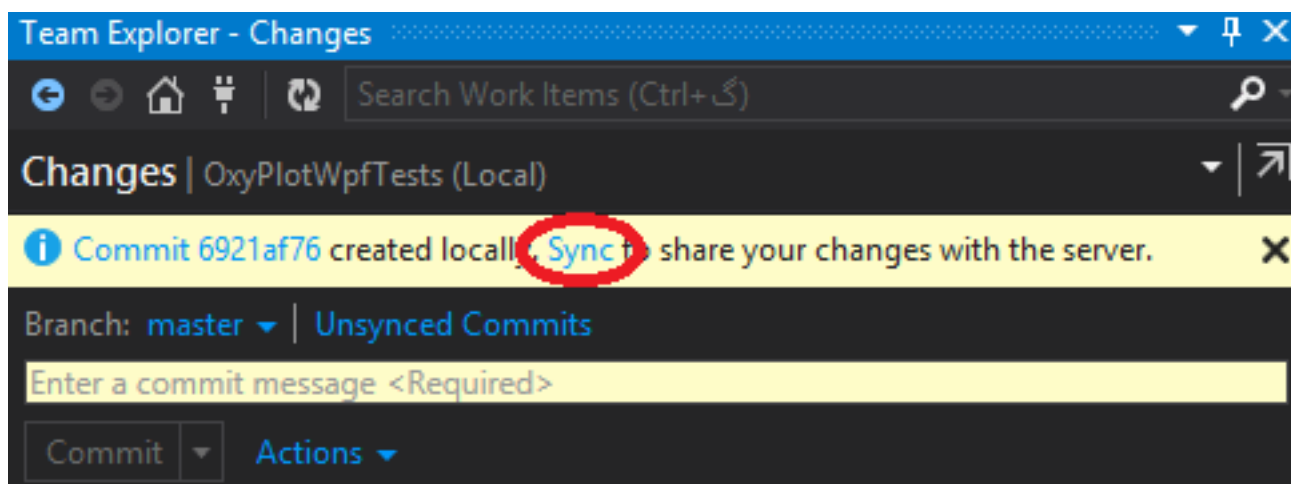
سپس نیاز است فایل‌های موجود را به مخزن کد GitHub ارسال کرد. بنابراین پس از مشاهده‌ی پیام موفقیت آمیز بودن عملیات همگام‌سازی، بر روی دکمه‌ی Home در نوار ابزار کلیک کرده و اینبار گزینه‌ی Changes را انتخاب کنید:



در اینجا پیام اولین ارسال را وارد کرده و سپس بر روی دکمه‌ی Commit کلیک کنید:



پس از مشاهده‌ی پیام موفقیت آمیز بودن commit محلی، نیاز است تا آنرا با سرور نیز هماهنگ کرد. به همین جهت در اینجا بر روی لینک Sync کلیک کرده و در صفحه‌ی بعدی بر روی دکمه‌ی Sync کلیک کنید:



اندکی صبر کنید تا فایل‌ها به سرور ارسال شوند. اکنون اگر به GitHub مراجعه کنید، فایل‌های ارسالی قابل مشاهده هستند:

An OxyPlot Sample — Edit

2 commits   1 branch   0 releases   1 contributor

branch: master   OxyPlotWpfTests / +

Init.

VahidN authored 3 minutes ago   latest commit 6921af7608

OxyPlotWpfTests	Init.	3 minutes ago
.gitattributes	Initial commit to add default .gitignore and .gitAttribute files.	18 minutes ago
.gitignore	Initial commit to add default .gitignore and .gitAttribute files.	18 minutes ago
OxyPlotWpfTests.sln	Init.	3 minutes ago

We recommend adding a README to this repository to help give people an overview of your project.   Add a README

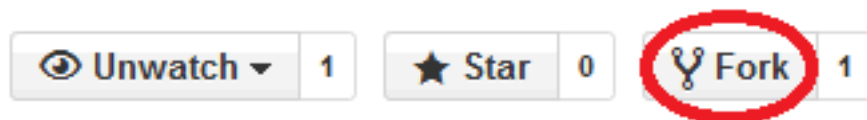
### اعمال تغییرات بر روی پروژه‌ی محلی و ارسال به سرور

در ادامه می‌خواهیم دو فایل README.md و LICENSE.md را به پروژه اضافه کنیم. پس از افزودن آن‌ها، یا هر تغییر دیگری در پروژه، اینبار برای ارسال تغییرات به سرور، تنها کافی است به برگه‌ی Team explorer مراجعه کرده و ابتدا بر روی دکمه‌ی Home کلیک کرد تا منوی انتخاب گزینه‌های آن ظاهر شود. در اینجا تنها کافی است گزینه‌ی Changes را انتخاب و دقیقاً همان مراحل عنوان شده‌ی پیشین را تکرار کرد. ابتدا ورود پیام Commit و سپس Commit. در ادامه Sync محلی و سپس Sync با سرور.

فرض کنید برای رفع باگی در پروژه‌ای از GitHub، ایده‌ای دارید. روند کاری اعلام آن، روش‌های مختلفی می‌تواند داشته باشند؛ از باز کردن یک Issue جدید تا فرستادن یک فایل zip و غیره. اما روش استاندارد مشارکت در پروژه‌های Git، ارسال یک PR یا Pull Request است. در ادامه نحوه‌ی انجام این‌کار را به کمک امکانات توکار VS.NET بررسی خواهیم کرد.

### ایجاد یک Fork جدید در GitHub

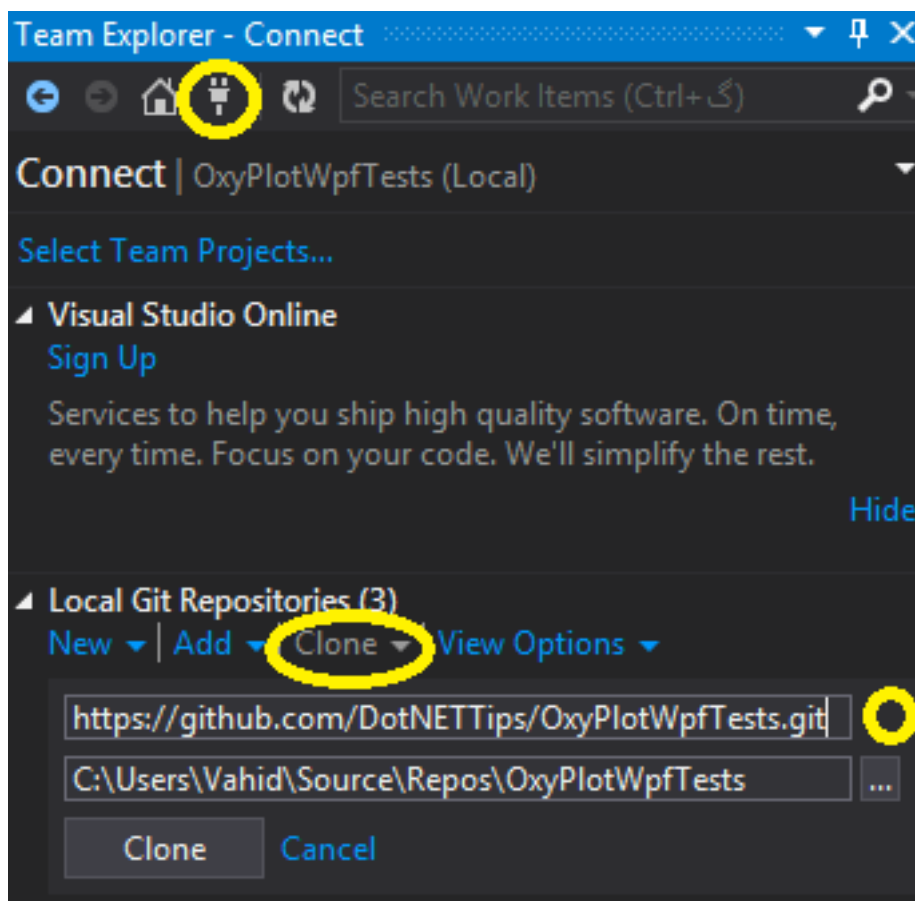
برای ارسال تغییرات انجام شده بر روی یک پروژه، نیاز است به صاحب یا مسئول آن مخزن در GitHub مراجعه و سپس درخواست دسترسی اعمال تغییرات را نمود. در این حالت، احتمال اینکه جواب منفی دریافت کنید، بسیار زیاد است. جهت مدیریت یک چنین مواردی، قابلیت به نام ایجاد یک Fork پیش بینی شده‌است.



در بالای هر مخزن کد در GitHub، یک دکمه به نام Fork موجود است. بر روی آن که کلیک کنید، یک کپی از آن پروژه را به مجموعه‌ی مخزن‌های کد شما در GitHub اضافه می‌کند. بدیهی است در این حالت، مجوز ارسال تغییرات خود را به GitHub و در اکانت خود خواهید داشت. نحوه‌ی اطلاع رسانی این تغییرات به صاحب اصلی این مخزن کد، ارسال همان PR یا Pull Request است.

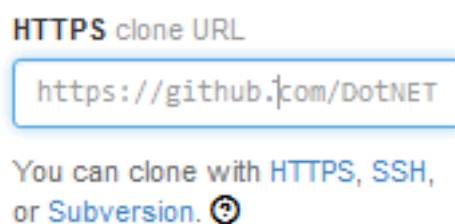
### دریافت مخزن کد Fork شده از GitHub به کمک Visual Studio

پس از اینکه Fork جدیدی را از پروژه‌ای موجود ایجاد کردیم، نیاز است یک Clone یا کپی مطابق اصل آن را جهت اعمال تغییرات محلی، تهیه کنیم. برای اینکار VS.NET را گشوده و به برگه‌ی Team Explorer آن که در کنار Solution Explorer قرار دارد، مراجعه کنید.



در اینجا بر روی دکمه‌ی Connect در نوار ابزار آن، کلیک کرده و در صفحه‌ی باز شده، بر روی لینک Clone کلیک نمایید. در اینجا می‌توان آدرس مخزن که Fork شده را جهت تهیه یک Clone مشخص کرد؛ به همراه محلی که قرار است این Clone در آن ذخیره شود.

آدرس HTTPS وارد شده، در کنار تمام مخازن که GitHub قابل مشاهده هستند:

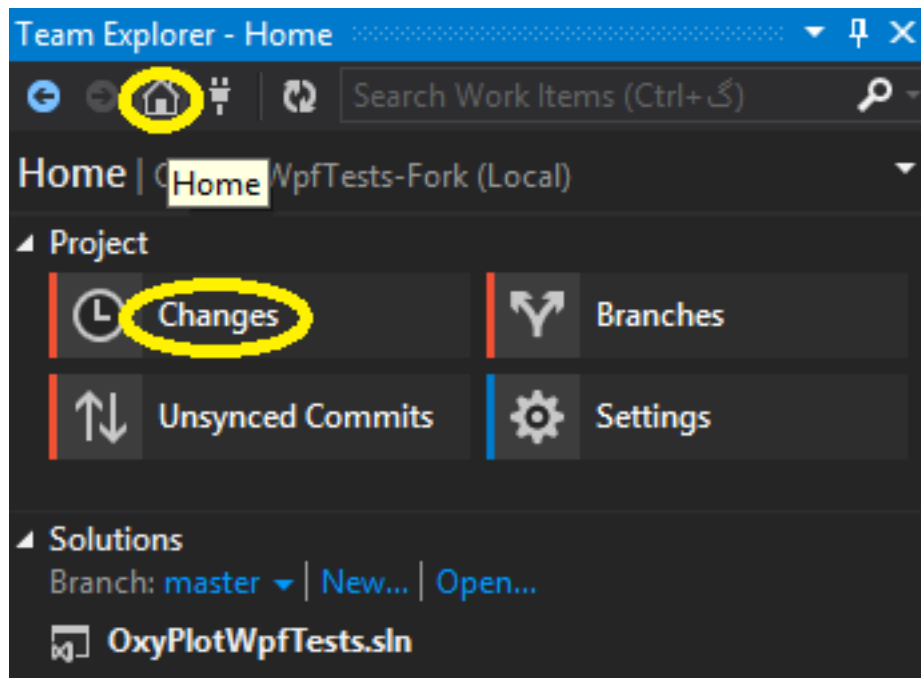


پس از تکمیل این دو آدرس، بر روی دکمه‌ی Clone کلیک نمایید. پس از پایان کار، اگر به آدرس محلی داده شده بر روی کامپیوتر خود مراجعه کنید، یک کپی از فایل‌های این مخزن، قابل مشاهده هستند.

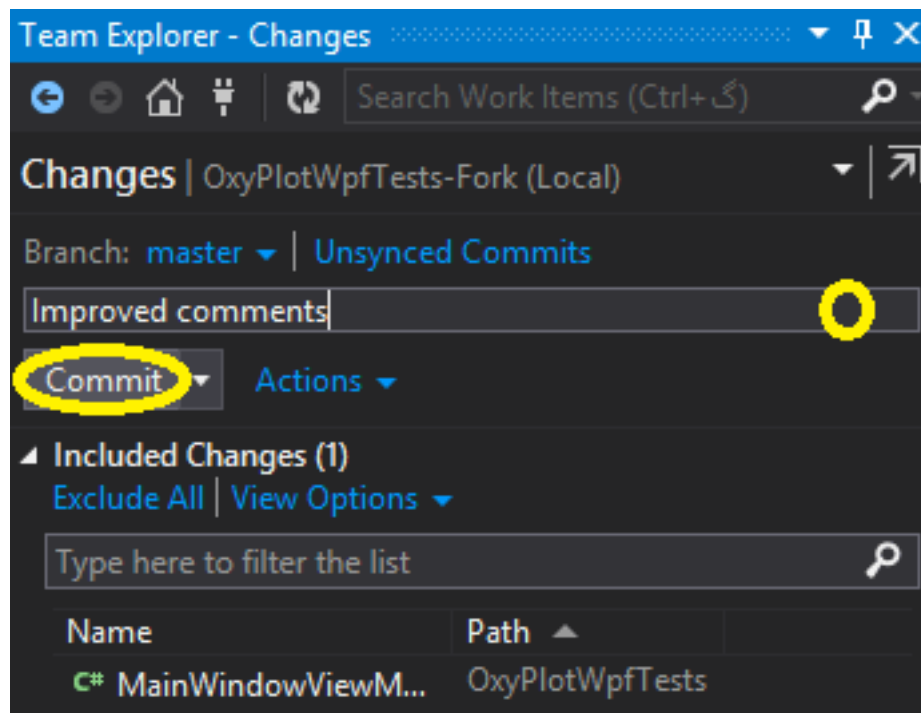
#### اعمال تغییرات محلی و ارسال آن به سرور GitHub

در ادامه، این پروژه‌ی جدید را در VS.NET باز کرده و تغییرات خود را اعمال کنید. اکنون نوبت به ارسال این تغییرات به سرور GitHub است. برای این منظور به برگه‌ی Team Explorer مراجعه کرده و بر روی دکمه‌ی Home آن کلیک کنید. سپس گزینه‌ی

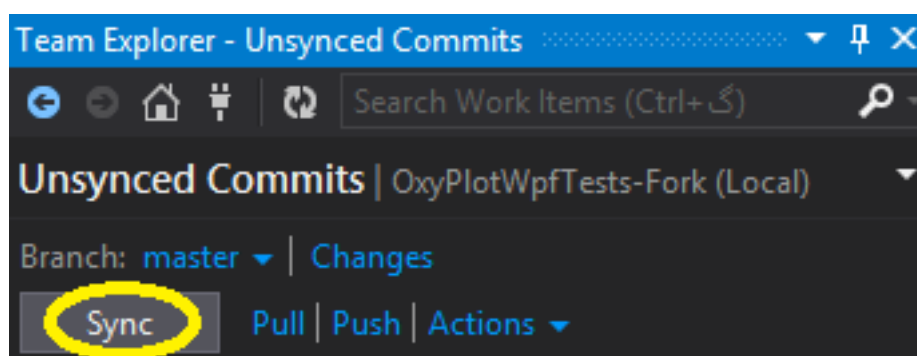
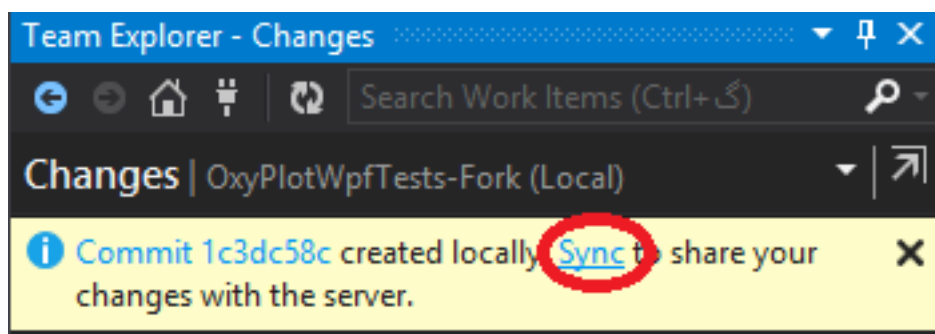
Changes را انتخاب نمایید:



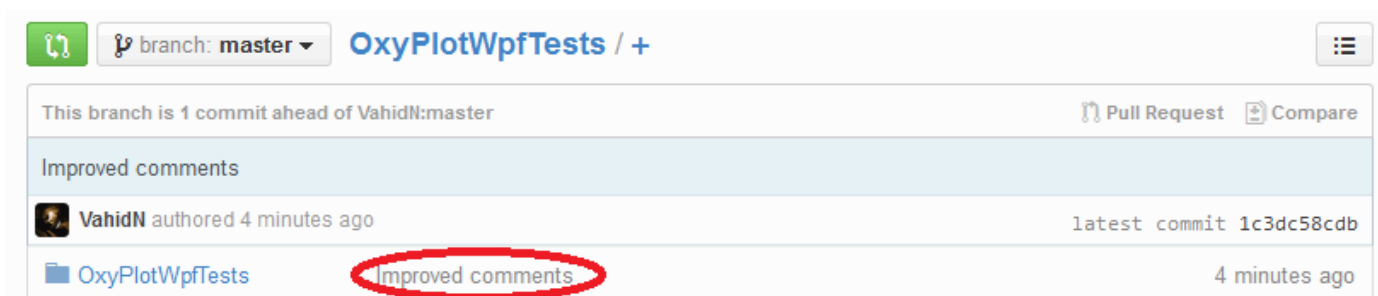
در اینجا توضیحاتی را نوشته و سپس بر روی دکمه‌ی Commit کلیک کنید.



پس از هماهنگ‌سازی محلی، اکنون نوبت به هماهنگ‌سازی این تغییرات با مخزن کد GitHub است. بنابراین بر روی لینک Sync در پیام ظاهر شده کلیک کنید و در صفحه‌ی بعدی نیز بر روی دکمه‌ی Sync کلیک نمایید:

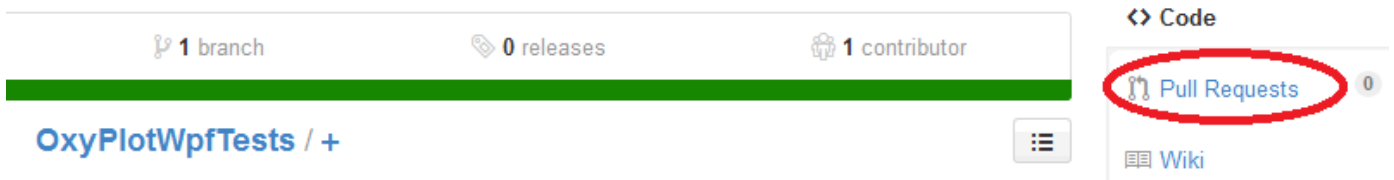


اکنون اگر به پروژه‌ی GitHub خود مراجعه کنید، این تغییر جدید قابل مشاهده‌است:



مطلع‌سازی صاحب اصلی مخزن کد از تغییرات انجام شده

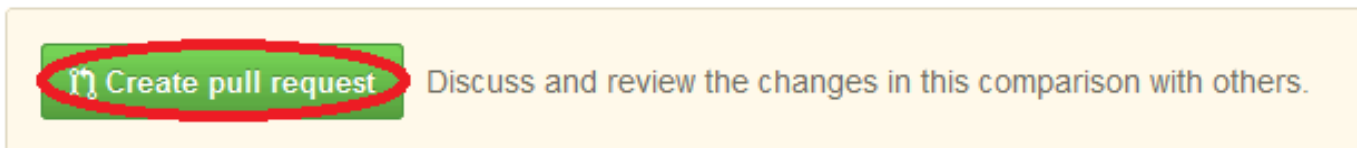
تا اینجا کسی از تغییرات جدید انجام شده‌ی توسط ما باخبر نیست. برای اطلاع‌رسانی در مورد این تغییرات، به مخزن کد Fork شده که اکنون تغییرات جدید به آن ارسال شده‌اند، مراجعه کنید. سپس در کنار صفحه بر روی لینک Pull request کلیک نمایید:



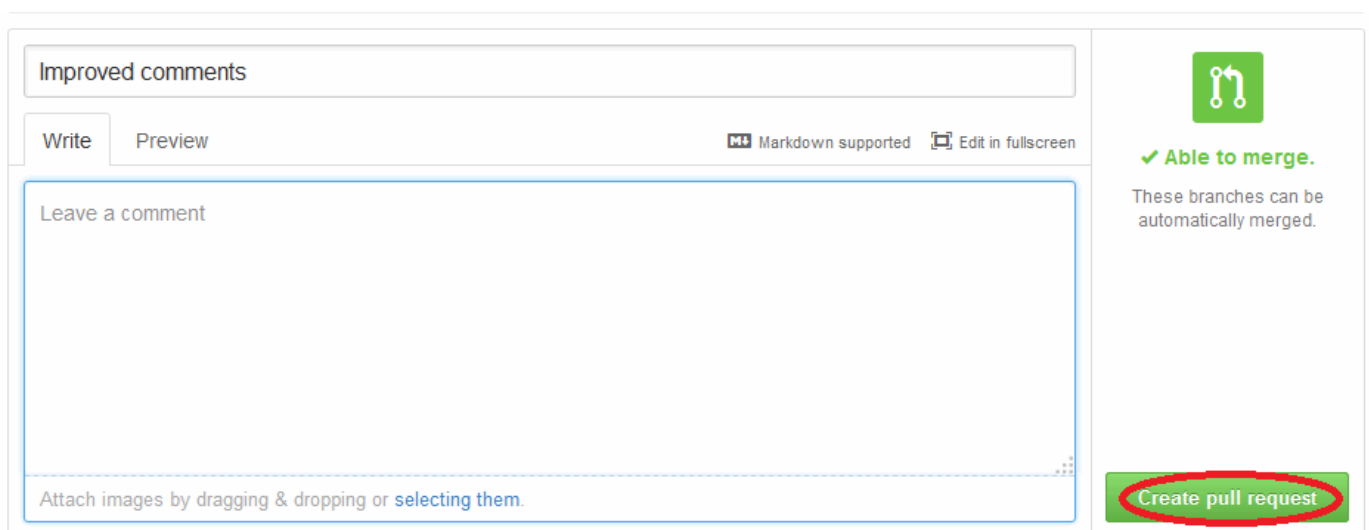
در اینجا بر روی دکمه‌ی New pull request کلیک کنید:



در ادامه تغییرات ارسال شما نمایش داده خواهند شد. آن‌ها را بررسی کرده و مجدداً بر روی دکمه‌ی Create pull request کلیک کنید:



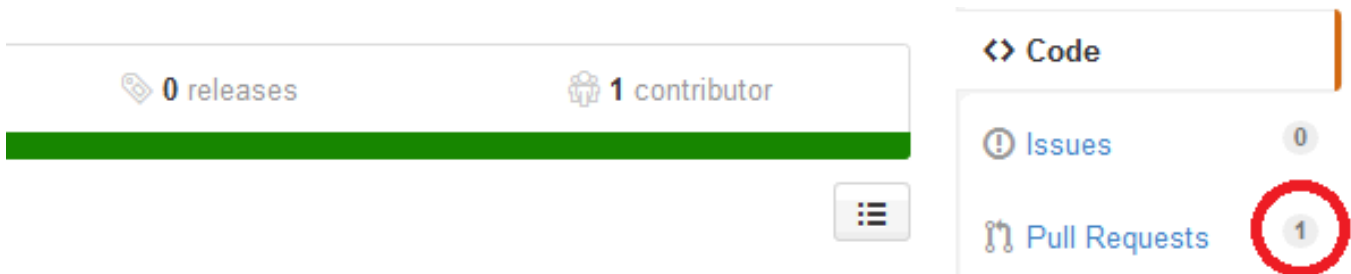
در اینجا عنوان و توضیحاتی را وارد کرده و سپس بر روی دکمه‌ی Create pull request کلیک نمایید:





## یکی سازی تغییرات با مخزن اصلی

اکنون صاحب اصلی مخزن کد یک ایمیل را دریافت خواهد کرد؛ همچنین اگر به مخزن کد خود مراجعه نماید، آمار Pull requests دریافتی قابل مشاهده است:

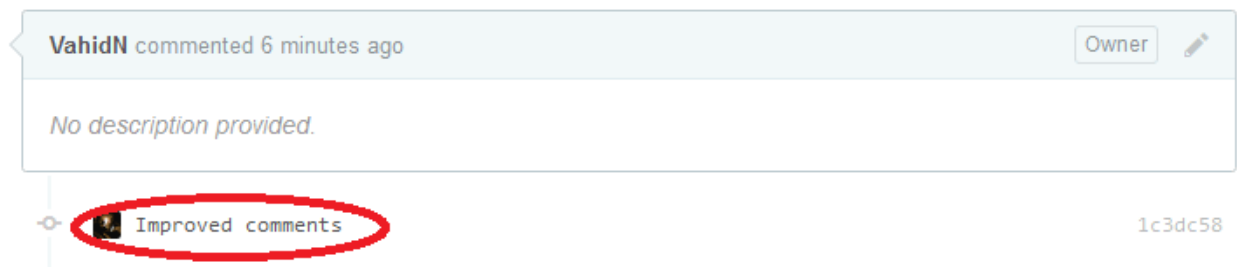


پس از انتخاب یکی از آن‌ها، لینکی برای بررسی تغییرات انجام شده و همچنین دکمه‌ای برای یکی سازی آن‌ها با پروژه‌ی اصلی وجود دارد:

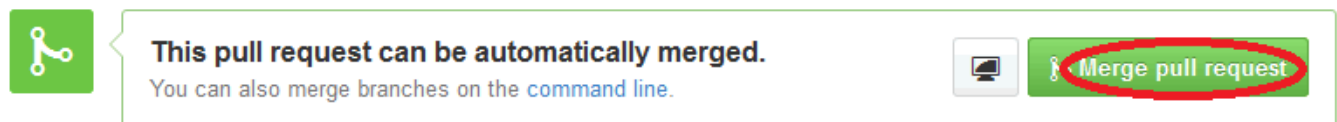
## Improved comments #1

**Open** VahidN wants to merge 1 commit into `VahidN:master` from `DotNETTips:master`

Conversation 0 Commits 1 Files changed 1

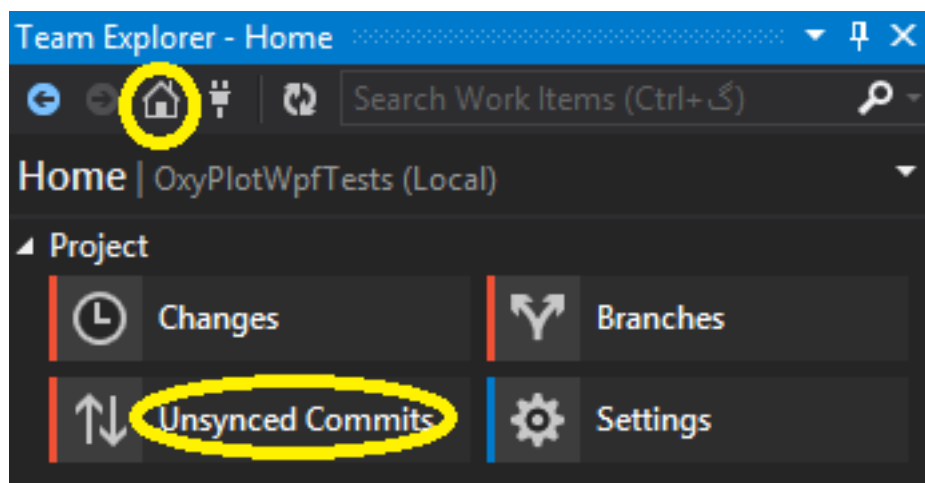


Add more commits by pushing to the **master** branch on **DotNETTips/OxyPlotWpfTests**.



## دریافت این تغییرات در مخزن کد محلی توسط صاحب اصلی پروژه

اکنون که این تغییرات با پروژه‌ی اصلی Merge و یکی شده‌اند، صاحب اصلی پروژه جهت تهیه‌ی یک کپی محلی و بهبود یا تغییر آن‌ها می‌تواند به صورت ذیل عمل کند:



ابتدا به برگه‌ی Team explorer مراجعه کرده و بر روی دکمه‌ی Home آن کلیک کنید. سپس گزینه‌ی Unsynced commits را انتخاب نمایید. در صفحه‌ی باز شده بر روی دکمه‌ی Sync کلیک نمایید. به این ترتیب آخرین تغییرات را از مخزن کد GitHub به صورت خودکار دریافت خواهید کرد:

