عنوان: **#3 آموزش سیستم مدیریت کد Git**
نویسنده: حسام امامی
تاریخ: ۱۳۹۱/۰۵/۱۶ ۱۴:۵۲
آدرس: www.dotnettips.info
برچسب‌ها: Git

شاید از دید بسیاری از برنامه نویسان بررسی نحوه عملکرد Git چندان اهمیتی نداشته باشد، زیرا آن‌ها سیستمی کارا برای مدیریت کدهای خود لازم دارند و نیازی نمی‌بینند که به جزئیات رفتار Git توجه کنند؛ به همین دلیل در بسیاری از منابع آموزشی این مفاهیم به این شکل گردآوری نشده است. اما من ترجیح دادم برای مدیریت و استفاده بهتر از Git حتی الامکان مطالب کاربردی را از پشت صحنه عملکرد Git در این بخش قرار دهم.

Working Tree (Directory) : پوشه‌ی روتی است که فایل‌های پروژه در آن نگه‌داری می‌شود. این پوشه باید حاوی پوشه‌ای به نام .git باشد که محتویات این پوشه، در اصل Repository ما را تشکیل می‌دهند.

اشیا در Git:

برای درک بهتر عملکرد سیستم مدیریت کد Git بهتر است نگاهی به اجزای تشکیل دهنده آن داشته باشیم. به طور کلی Git دارای 4 نوع object است، که هر کدام وظیفه خاصی را به عهده دارند:

1)Tree: شیئ Tree دقیقا مانند دایرکتوری‌ها در یک سیستم مدیریت فایل است. در واقع treeها ساختاری درختی را ایجاد می‌کنند تا وضعیت فایل‌ها و پوشه‌ها را در Repository حفظ نمایند. هر tree توسط یک کد منحصر به فرد SHA-1 نام گذاری می‌شود.

2)BLOB(Binary large object): اگر با سیستم‌های مدیریت داده نظیر SQL Server کار کرده باشید قطعا با BLOB آشنایی دارید. BLOBها در واقع چیزی نیستند جز یک مجموعه از بایت‌ها که می‌توانند حاوی هر چیزی باشند (نظیر عکس، فایل متنی، فایلهای اجرایی و...) در Git فایل‌ها به صورت BLOB و به شکل کامل ذخیره می‌شوند. همچنین مقدار هش شده محتویات فایل‌ها با استفاده از SHA-1 در خود فایل ذخیره می‌شود. به این ترتیب در صورت تغییر در فایل، مقدار هش جدید با مقدار موجود در فایل فرق کرده و Git متوجه می‌شود که فایل دچار تغییر شده است. نکته قابل توجه این است که بر خلاف بسیاری از سیستم‌های مدیریت کد، در هر بار تغییر فایل، Git تنها تغییرات را ذخیره نمی‌کند بلکه از کل محتوا یک snapshot می‌گیرد. شاید به نظر بسیاری تهیه این snapshotهای فراوان باعث زیاد شدن حجم Repository شود، اما Git هوشمندانه تنها فایل‌هایی را مجددا ذخیره می‌نماید که مقدار آن‌ها تغییر کرده است. در غیر این صورت یک نشانگر به فایل موجود در snapshot جدید ایجاد می‌کند.

3)Commit: این شئ، یک snapshot از وضعیت فعلی Working Tree است. در واقع با هر با دستور commit این object ایجاد شده و حداقل حاوی اطلاعات زیر است:

مقدار هش درختی که به آن اشاره می‌کند

نام ثبت کننده دستور commit

توضیحی درباره علت ایجاد commit

خود commit نیز توسط یک کد منحصر به فرد SHA-1 شناخته می‌شود.

4)Tag : چون کار کردن با کدهای هش commit ممکن مشکل باشد، می‌توان از تگ‌ها به عنوان نامی برای commit استفاده نمود. خود تگ می‌تواند حاوی توضیحاتی باشد.

آشنایی با Stage(Index):

هر فایل قبل از آنکه بخواهد در Repository توسط دستور commit ذخیره شود باید ابتدا به Stage آورده شود. در این حالت Git تغیرات فایل را دنبال کرده و سپس می‌توان توسط دستور commit فایل را در Repository وارد کرد. بنابراین ذخیره یک فایل در Git دارای سه مرحله است:

Modified : یعنی فایل تغییر کرده اما به stage اضافه نشده است

Staged: فایل تغییر کرده به stage اضافه شده است.

Commited: فایل در Repository ذخیره شده است.

:head

اشاره‌گری است که به آخرین شئ commit اشاره می‌کند. هر Repository می‌تواند یک head برای هر شاخه‌ی مختلف داشته باشد؛ اما در هر لحظه تنها یک head به عنوان head جاری شناخته می‌شود که معمولا آن را با حروف بزرگ یعنی HEAD مشخص می‌کنند.

تا این مرحله شما تقریبا تمامی آنچه که برای شروع کار با Git را لازم دارید آموخته‌اید. البته همان‌طور که در ابتدا اشاره کردم این مباحث دارای جزئیات بسیاری است اما تا این اندازه برای کار با Git کفایت می‌کند. در صورتیکه به نکات خاصی احتیاج پیدا کنیم، در طول بیان دستورات Git به آن‌ها اشاره خواهد شد. در قسمت بعد نحوه‌ی نصب و پیکره‌بندی Git را بررسی می‌کنیم.

نظرات خوانندگان

نویسنده: رضا
تاریخ: ۱۳۹۱/۰۵/۱۶ ۲۱:۲

من در مورد ترتیب modified و stage شک دارم .
وقتی یک فایل به پوشه پروژه اضافه میشه برای اینکه تغییراتش توسط Git دنبال و ثبت بشه باید وارد stage بشه یا به عبارتی Add بشه و در اولین commit بعد از اون به عنوان staged ثبت میشه . از این به بعد تغییرات در این فایل دنبال و در commitهای بعدی به عنوان modified نشون داده میشه . درسته ؟ یا اشتباه متوجه شدم ؟

نویسنده: حسام امامی
تاریخ: ۱۳۹۱/۰۵/۱۶ ۲۱:۵۴

خیر به این صورت نیست تصور کنید شما پنج فایل درون working directory خود دارید همچنین دو فایل جدید نیز اضافه کردید تا زمانی که آن‌ها را با استفاده از دستور add به stage نیاورید git اقدامی برای ساخت سابقه برای آن فایل‌ها نمی‌کند به عنوان مثال سه فایل از پنج فایلی که قبلا وجود داشته تغییر کرده باشد و از این سه فایل تغییر کرده تنها دو تا و یکی از فایل‌های جدید به stage اضافه شده شده باشند و دستور commit اجرا شود تنها همان دو فایل تغییر کرده و فایل جدید موجود در stage در repository ذخیره می‌شوند
اما در مورد سوال شما می‌تونید فعلا به این صورت تصور کنید که بعد از commit فایل از روی stage حذف میشه (البته دستورات git در این زمینه متفاوت عمل می‌کنند و لزوما اینگونه نیست) بنابراین فایلی که قبلا commit شده و الان تغییر کرده و روی stage نیست وضعیت modified دارد

نویسنده: هوشنگ
تاریخ: ۱۳۹۱/۰۵/۱۸ ۰:۲۲

بیصبرانه منتظر قسمت‌های بعدی اون هستم . میخوام هر چه سریعتر به قسمت GitExtension و Git Source Control Provider برسیم . کلی سوال واسم ایجاد شده .