Manual för Crystal och Git

Richard Simko

24 januari 2014

Innehåll

1 Git

Git är ett konfigurationshanteringssystem precis som Subversion (SVN) och CVS. Git skiljer sig dock på några sätt. Vissa av dessa kommer uppenbara sig under den här kursens gång då användandet av Crystal kommer kräva vissa av de något mer avancerade funktionerna i Git. Denna guide kommer förklara de skillnader som finns och hur man ska använda Git för att kunna utnyttja verktyget till dess fulla potential.

1.1 Lokala repositorys

Den viktigaste skillnaden mellan Git och SVN (I alla fall i den här kursen) är att Git använder sig av ett lokalt repository. Detta innebär att varje utvecklare har sin egen kopia av hela repositoryt, inklusive all historik. Detta innebär att det som i SVN motsvaras av `svn commit` kräver två kommandon i Git, något som kommer visas senare.

Crystal analyserar detta lokala repository för att sammanställa sin data, det är därför viktigt att alla utvecklare även utnyttjar sig av detta.

1.2 Hämta repositoryt (`git clone`)

Innan du kan hämta hem repositoryt från Git krävs det att du konfigurerar SSH-nycklar så att processen går så smidigt som möjligt. Hur

du gör detta finns beskrivet här: `https://help.github.com/articles/generating-ssh-keys`.

För att hämta Repositoryt använder du dig av kommandot `git clone` med ett argument, var repositoryt finns. Exempelvis, för att hämta kursens repository använder du dig av `git clone git@github.com:RichardSimko/EDA260-Team09.git`

1.3 Hämta ändringar (`git pull`)

OBS! Pga. Crystal kommer vi använda oss av ett alternativt sätt att göra commits. Detta heter `gitcom` och finns inlagt på de konton som kommer användas.

När ändringar skett i den centrala repositoryn (Kallad REMOTE) kan du hämta dessa ändringar med hjälp av kommandot `git pull`. Detta motsvarar `svn update` och hämtar ner ändringarna och försöker mergea dem med den kod du har lokalt. Detta måste göras innan man kan skicka in sina ändringar (Git kommer påminna dig om detta om du glömmer det).

1.4 Skicka in ändringar (`git commit -a` och `git push`)

Som nämnts tidigare måste man först skicka in ändringarna man gjort till sitt lokala repository. Detta görs med hjälp av `git commit -a` (`-a` betyder att alla ändringar ska commitas). När du exekverar detta kommando kommer du presenteras med din default texteditor där du ombeds skriva in ditt commitmeddelande. När du stänger editorn skickar Git in din commit.

Det kan dock hända att Git säger att du har filer som inte är tillagda (`nothing added to commit but untracked files present (use ‘‘git add’’ to track)`). Denna varning innebär att alla ändringar du gjort finns i filer som inte är tillagda i Git. Därför måste man alltid, när man skapar en fil, ställa sig i root-mappen för workspacet och köra följande kommando `git add .` (Med “.” på slutet). Detta lägger till alla filer som inte redan var tillagda till Git. Därefter kan man göra `git commit -a` igen.

Slutligen, för att andra de utvecklarna ska kunna ta del av den kod du producerat krävs det att du “pushar” den till Github. Detta görs med hjälp av `git push`.

1.5 Vanliga problem

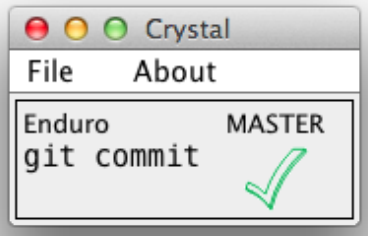
Det här avsnittet kommer fyllas i under kursens gng...

2 Crystal

2.1 Installation

Alla filer som behövs för Crystal finns i repositoryt under `/Crystal/`.

1. Öppna `conflictClient.xml` och ange ditt användarnamn samt vart ditt repository finns på de platser detta behövs
2. Skriv följande i terminalen: `mv conflictClient.xml ~/.conflictClient.xml`
3. Starta Crystal och verifiera att allt ser ut som i Figur 1



Figur 1: Första startskärmen

2.1.1 Lägga till fler workspaces

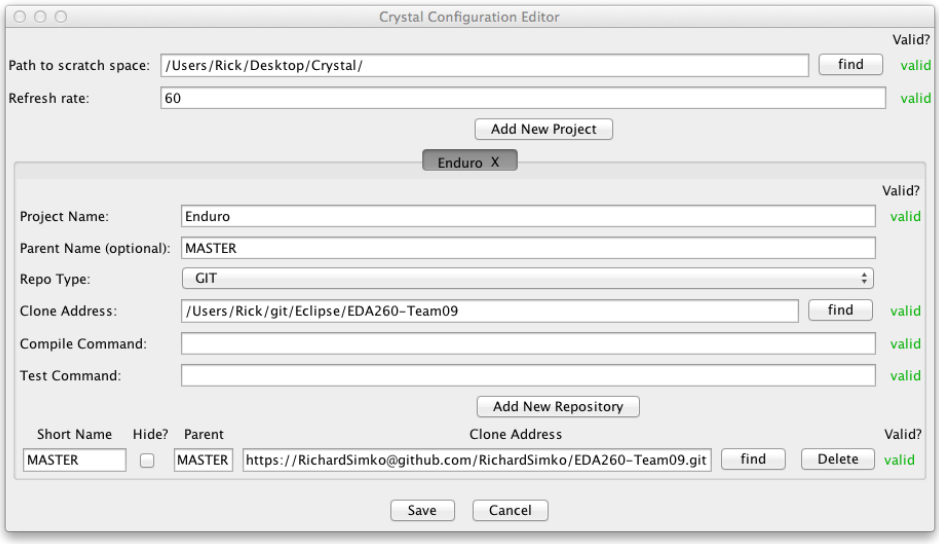
Crystal fungerar bättre ju fler utvecklares workspaces finns inlagda, därför är det viktigt att ni delar med er av varandras workspaces. Detta gör man enklast på följande vis.

1. Öppna en terminal i den mapp (T.ex. om ditt workspace ligger i `~/EDA260-Team09/` ska du stå i den mappen)
2. Skriv följande kommando `chmod -R a+r ./`

Detta ger andra användare läsrättigheter till den mapp där du skriver kommandot samt alla undermappar. Det är därför viktigt att du gör detta i rätt mapp, annars ger du övriga användare läsrättigheter till saker de inte borde komma åt.

När alla användare gjort detta kan ni lägga till varandras workspaces i Crystal. Detta gör man på följande vis: (Se även Figur 2):

1. Gå in under File -> Edit configuration i Crystal
2. Klicka på "Add new Repository
3. Ange namnet på repositoryt (Ex. den andra användarens användarnamn)
4. Ange parent som "MASTER"



Figur 2: Första startskärmen

5. Ange sökvägen till den andra utvecklarens repository (T.ex. `/h/d3/k/dt09rs6/EDA260-Team09`)

Det bör se ut som i Figur 3, det bör även stå valid i grön text längst ut till höger. När du sparar kommer programmet meddela dig om det finns några fel. Upprepa sedan denna processen för varje ny utvecklare.

Figur 3: Korrekt workspace-inställning

2.2 Användning

Crystal har en någorlunda läslig manual tillgänglig på internet.[1], den kan dock vara svårläst utan vissa förklaringar som följer här. Det kommer göras några referenser till Crystals användarmanual så läsaren uppmanas att ha båda lätt tillgängliga.

2.2.1 Relationships

Då vi fann detta något otydligt kommer vi här att förklara detta i något större detalj än vad Crystals manual gör.

Crystal använder sig av ett antal olika symboler för att illustrera hur ditt workspace förhåller sig till andras. Utöver detta kan även de olika symbolerna

[1] `https://code.google.com/p/crystalvc/wiki/CrystalUserManual`

vara färgade olika för att indikera om du kan genomföra någon åtgärd för att förändra förhållandet mellan ditt workspace och någon annans.

Under rubriken MASTER anges förhållandet mellan ditt workspace och huvudrepositoryt (Det som ligger på Github). De olika symbolerna följer här:

Bock eller SAME Dessa betyder att ditt repository och den andra utvecklarens är i synk. Ni har samma version av samma filer och ingen skillnad finns.

Uppil eller AHEAD Detta betyder att ditt repository har nyare filer än den andra utvecklarens. Du har alltså commitat saker som den andra inte har. I den här situationen kan det vara bra att göra en `git push` och meddela de andra.

Nedpil eller BEHIND Någon annan har ändringar som inte du har, om denna syns på MASTER kan det vara bra att göra en `git pull` för att få ner de senaste förändringarna.

Gul branch eller MERGE Både du och den andra utvecklaren har gjort ändringar men dessa kan mergas automatiskt. I den här situationen kan det vara bra att göra en `gil pull` följt av `git push`

Röd branch eller TEXTUAL_X Detta innebär att en konflikt kommer att uppstå mellan dig och den andra utvecklaren, ni har alltså ändrat på samma ställe i samma fil. I den här situationen kan det vara bra att försöka göra en merge så snart som möjligt för att minska konfliktens omfattning.