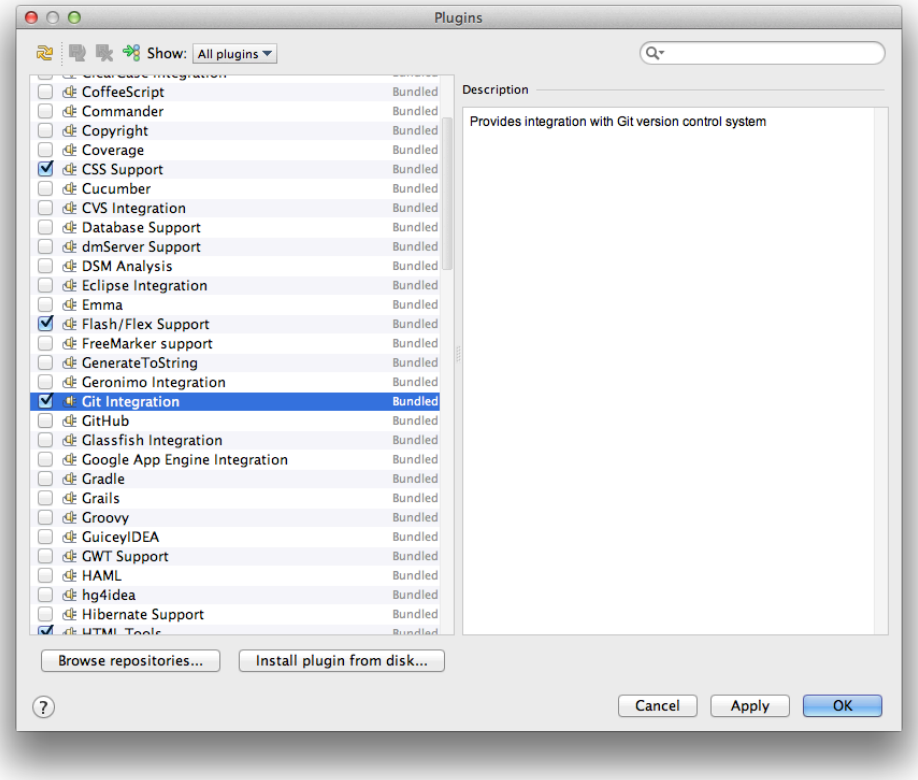
GIT

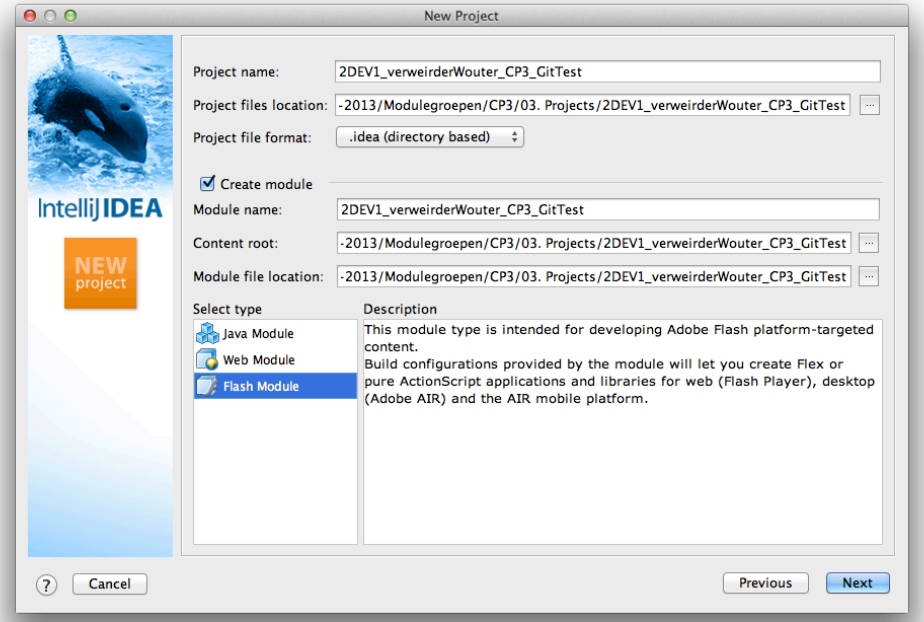
GIT is een versie-beheer systeem ontwikkeld door Linus Torvalds, waarmee je onafhankelijk van een server historiek van files & versies kunt managen. Je kunt een GIT repository ook op een centraal toegankelijke plaats delen om met meerdere ontwikkelaars samen code te synchronizeren.

GIT IN INTELLIJ

IntelliJ komt met ondersteuning voor GIT repositories. Mogelijk moet je die ondersteuning nog activeren bij je plugins. Open de plugin manager (link in het startvenster van IntelliJ), en zorg dat zowel de GIT plugin als Task Management aangevinkt zijn:

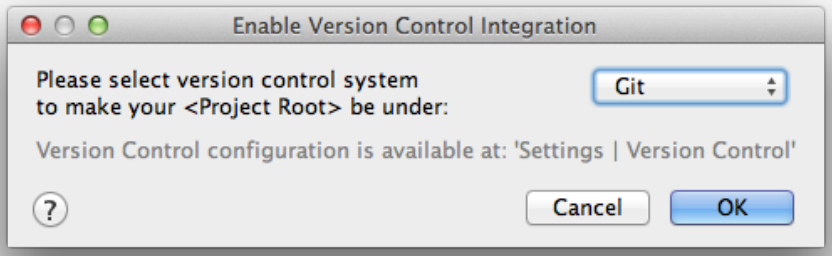


De GIT plugin zal standaard werken op project-niveau (niet op module niveau). Maak dus een volledig nieuw project aan in een lege directory, en gebruik volgende naamgeving: KLAS_familienaamVoornaam_CP3_GitTest. Maak ook een Flash module aan met dezelfde naam:

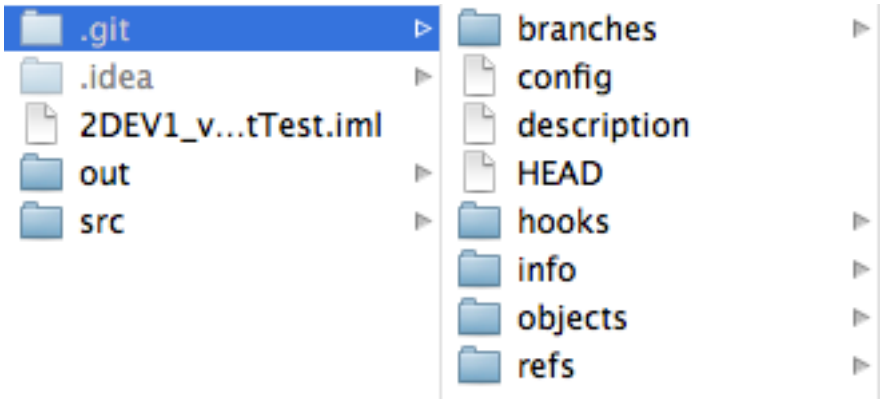


Kies ervoor om een pure actionscripct desktop application te maken en klik op Finish om het project & de module aan te maken.

Na het aanmaken van het project, moet je het project laten managen door het GIT systeem. In de menubalk kies je voor VCS, Enable Version Control Integration. Kies hier voor GIT.



Dit zal een hidden .git directory aanmaken in je project root. Je kunt hidden files / folders in je finder tonen/verbergen via het Secres Preference Pane (http://secrets.blacktree.com/). Zorg ervoor dat je die .git directory kunt raadplegen in finder en bekijk de structuur:



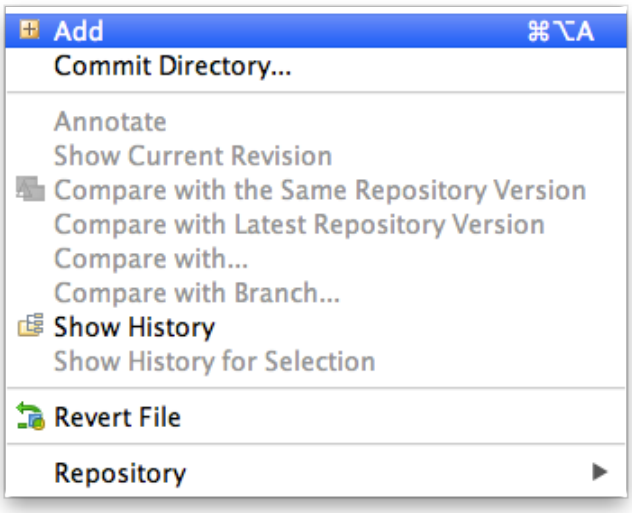
Zelf zul je haast nooit te maken krijgen met deze hidden directory: deze wordt gebruikt door het GIT systeem en is te vinden bij elke GIT repository. De hidden directory bevat alle informatie over de repository zoals geschiedenis van files, filenames, versies van je project, etc...

Bekijk even de inhoud van de objects map in die .git directory.De objects map zal alle informatie zoals inhoud van bestanden, namen van bestanden & directorystructuren bevatten. Je zal deze zien groeien naarmate je project evolueert. Momenteel is deze map (op de submappen info & pack na) leeg. Er wordt momenteel nog niets gemanaged door GIT.

BESTANDEN MANAGEN MET GIT

Klap in IntelliJ de src & idea mappen open. Je ziet dat de filenames een rode kleur hebben. Dit wil zeggen dat deze nog niet gekend zijn door GIT.

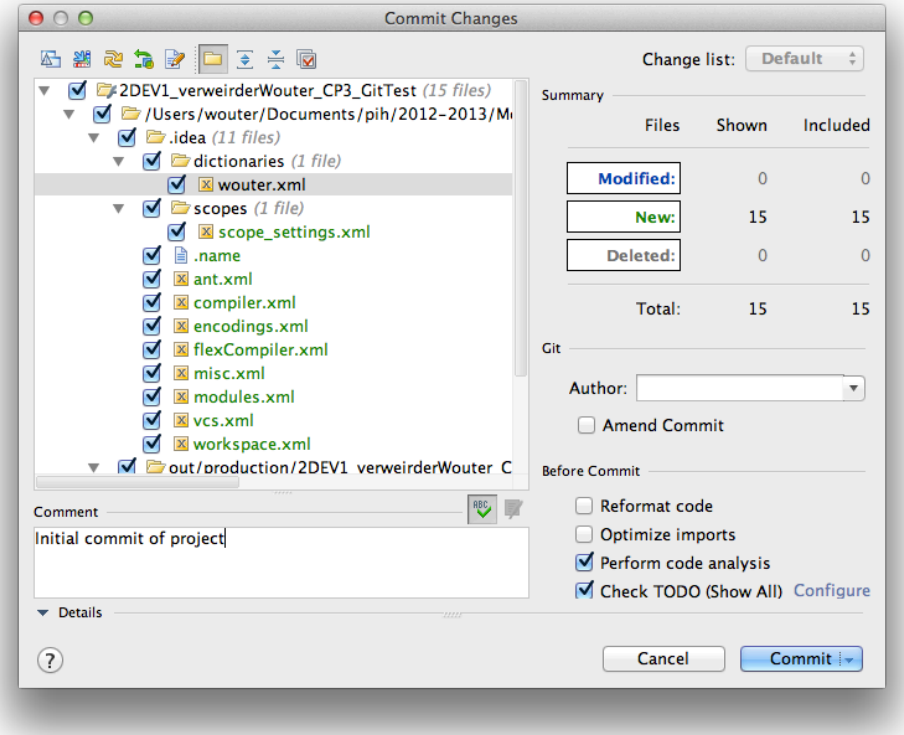
Klik rechts op de module in IntelliJ en kies in het GIT menu voor Add:



Je merkt dat de files nu een groene kleur hebben gekregen. Dit wil zeggen dat GIT van het bestaan van die bestanden afweet.

Bekijk opnieuw de objects map. Je ziet hier nu heel wat subfolders met bestanden met cryptische filenames. De filenames komen overeen met de SHA1-hash van de inhoud van de files.

Klik vervolgens in het GIT menu op Commit Directory… Dit opent een venster met enkele input-opties:



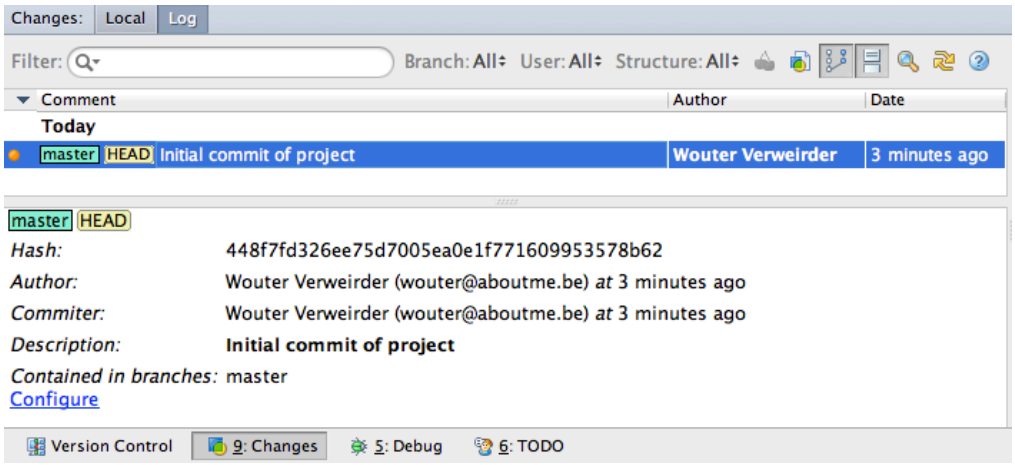
Een commit zal een versie van je project saven in de GIT repository. Telkens wanneer je een set van wijzigingen in je project wil bijhouden, zal je – na telkens een Add actie te doen – een commit uitvoeren. Je geeft elke commit wat commentaar, zodat je later makkelijk kan terugvinden wat er precies gewijzigd is in het project.

Kies voor Commit → commit om je projectstatus te saven in de repository. Je merkt dat de kleuren van de filenames nu zwart zijn in InteliJ.

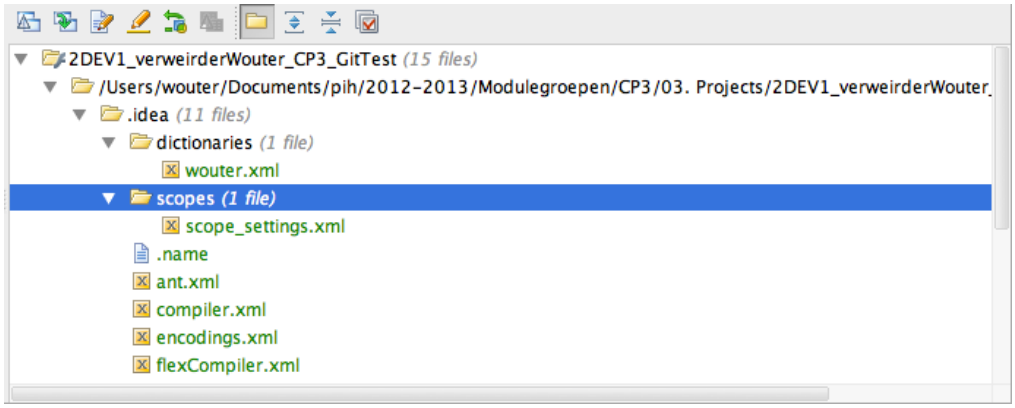
Je zal zien dat er in de .git folder nieuwe objecten worden aangemaakt, en dat er – onder andere – in de refs map een submap heads staat met daarin een file master. Deze file bevat informatie over welke versie de actieve versie is van je project.

CHANGES PANEL

Je zal vaak gebruik maken van het Changes Panel in IntelliJ. Dit kun je onderaan je venster vinden. Klik dit open en kies voor de Log view. Indien je nog niets ziet in de panel, klik je op de refresh pijltjes in dit panel om de GIT info opnieuw op te halen:



Hier zul je alle commits kunnen bekijken in je repository en zo een beeld krijgen van de historiek van je project. Wanneer je een commit aanklikt, verschijnt aan de rechterkant meer informatie & acties die te maken hebben met die bepaalde commit:



Dit zul je straks nog nodig hebben om onder andere verschillen tussen files / commits te bekijken.

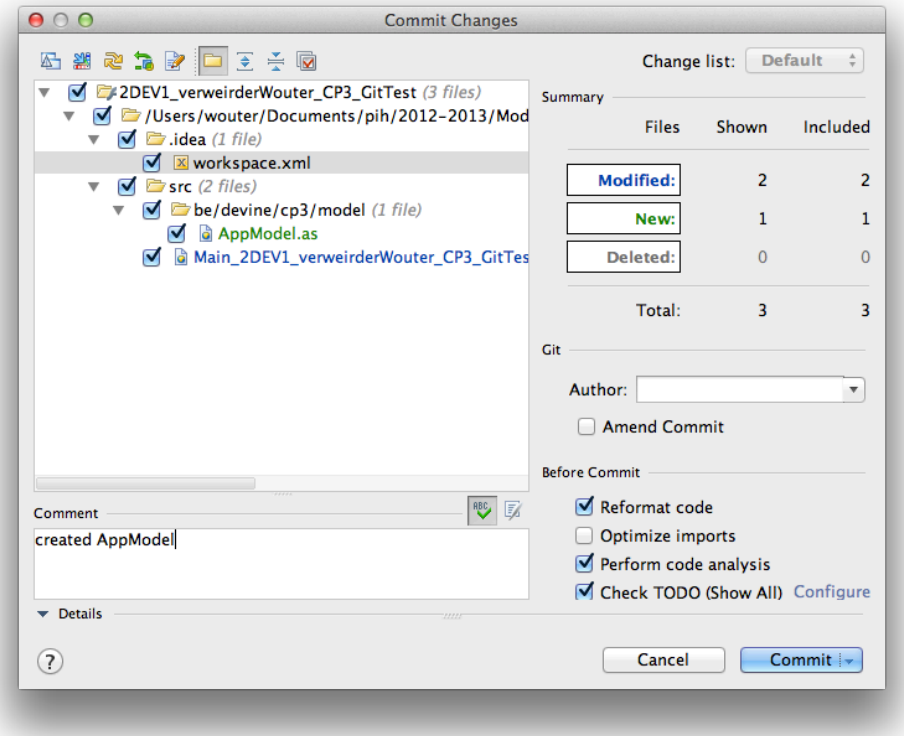
EEN TWEEDE COMMIT

Maak een nieuwe klasse be.devine.cp3.model.AppModel aan. Je krijgt meteen de vraag of je de nieuwe file wil adden. Kies voor Yes.

Maak daarna in je opstartklasse een instantie aan van deze AppModel klasse:

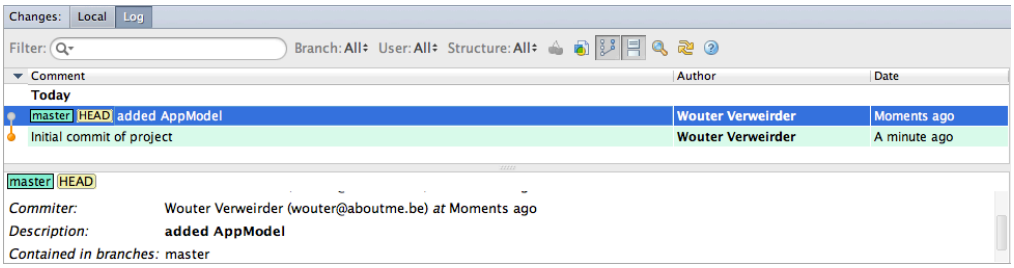
```
package {

import be.devine.cp3.model.AppModel;

import flash.display.Sprite;
import flash.text.TextField;

public class Main_2DEV1_verweirderWouter_CP3_GitTest extends Sprite {
    public function Main_2DEV1_verweirderWouter_CP3_GitTest() {
        var textField:TextField = new TextField();
        textField.text = "Hello, World";
        addChild(textField);

        var appModel:AppModel = new AppModel();
    }
}
}
```

Je zal zien dat het nieuwe bestand een groene kleur heeft, en je opstartklasse – die gewijzigd is – een blauwe kleur. Klik rechts op je Module, en kies in het GIT menu opnieuw voor Commit Directory:



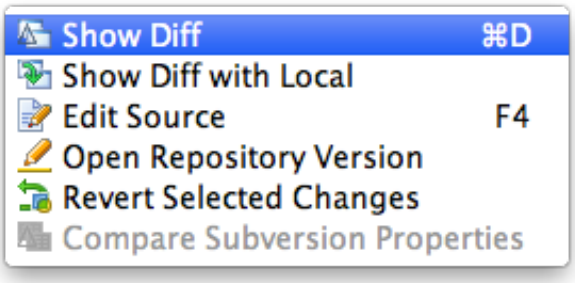
Als je een warning krijgt, mag je dit in ons geval negeren. Bij je project zelf is het een goed idee om deze warnings ter harte te nemen en problemen op te lossen.

Refresh het log panel onderaan, je zou je 2 commits in de lijst moeten zien staan:

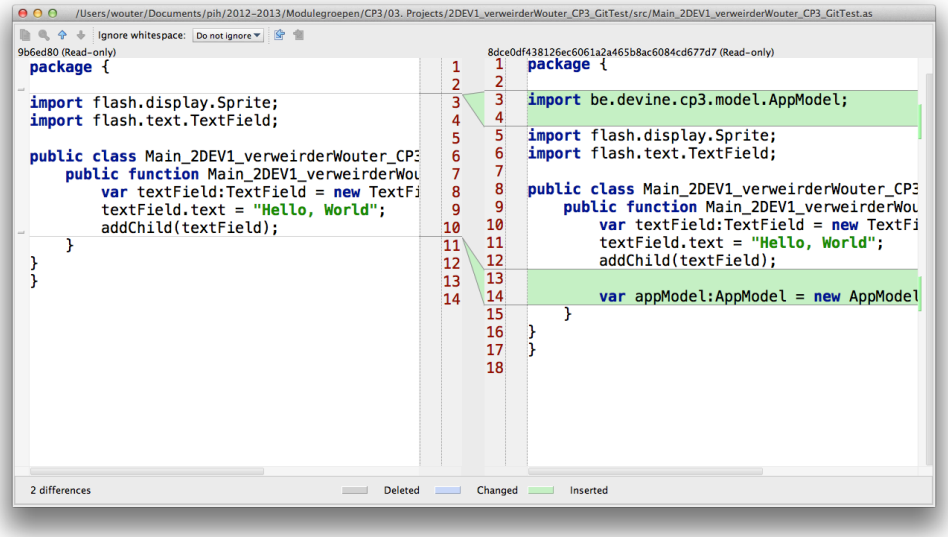


REVERTEN NAAR EEN VORIGE VERSIE

Je kunt op een overzichtelijke manier wijzigingen bekijken die gebeurd zijn. Selecteer in de laatste commit het opstartbestand (waarin je een instantie van AppModel aanmaakte). Klik rechts, en kies voor Show Diff...

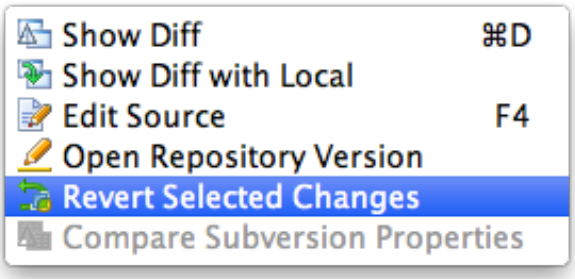


Je krijgt een venster te zien waarin de changes aangeduid worden.

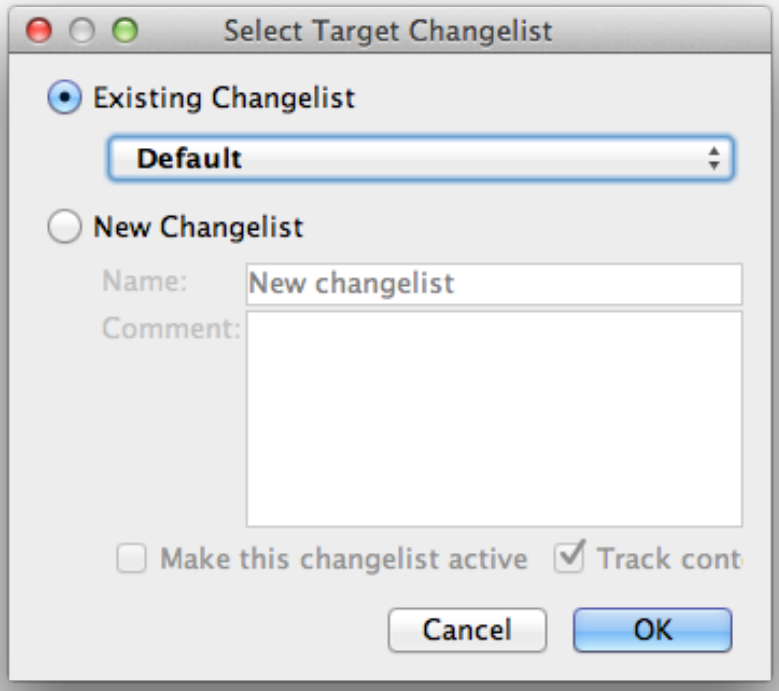


Stel dat je terug wil gaan naar een vorige versie in de repository. We zijn bijvoorbeeld niet blij met het aanmaken van het AppModel, en willen terugkeren naar de vorige versie:

Selecteer in de laatste commit het opstardbestand, klik rechts en kies voor "Revert Selected Changes".



In het changelist venster kies je voor Default, en klik je op OK om het bestand te wijzigen zodat de inhoud aangepast is naar de vorige versie:



Je ziet dat het opstartbestand een blauwe kleur heeft gekregen. GIT heeft de inhoud van de file aangepast naar de inhoud van een vorige versie. Dit is een wijziging die je nog moet committen!

Commit deze revert naar je repository en voorzie je commit van een beschrijvende comment.

LOKALE WIJZIGINGEN ONGEDAAN MAKEN

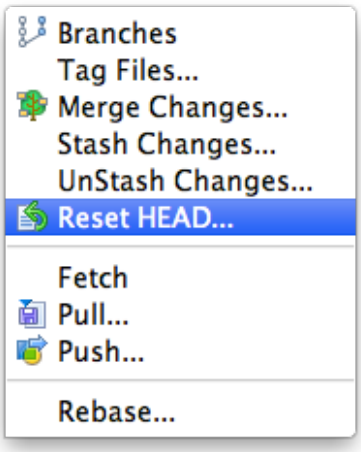
Stel dat je lokaal verschillende aanpassingen hebt gedaan die niet zo goed uitgedraaid zijn: er zijn een pak nieuwe bugs opgedoken, of het project compiled niet meer. Je hoeft niet file per file undo-commando's uit te voeren, maar kunt resetten naar de laatste commit in je repository.

Voer de volgende acties uit:

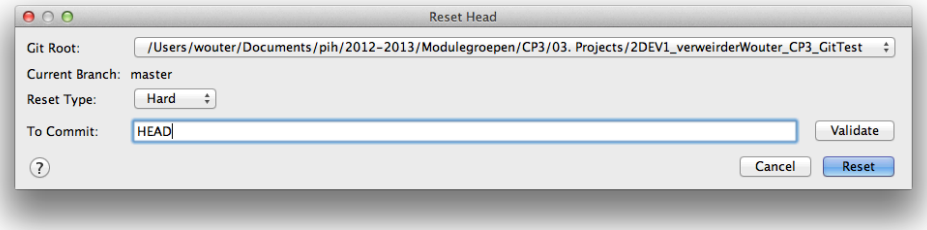
- Doe enkele wijzigingen in je opstartklasse en sla je klasse op (niet committen!)
- Wis de AppModel klasse
- Maak een nieuwe klasse be.devine.cp3.view.Image aan

We zijn niet tevreden met deze wijzigingen, en willen ons project terugzetten zoals deze laatste gecommit werd.

Klik rechts op je module en kies voor Git → Repository → Reset HEAD



Je krijgt een venster te zien waarin je het reset type kunt kiezen (Mixed, Soft of Hard) en een commit kunt specifiëren (elke commit heeft een uniek ID, er zijn ook aliassen zoals HEAD die de laatste commit betekent). Kies voor een Hard reset en klik op Reset:



Indien IntelliJ voorstelt om je project te reloaden, mag je dit doen. Je ziet dat het project gereset is naar de versie die je laatst gecommit hebt.

DELEN OP GITHUB

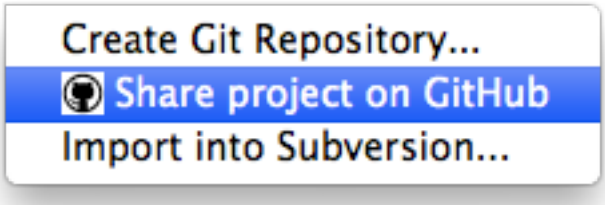
Het is de bedoeling dat je met meerdere mensen kunt samenwerken aan een project. Dit kun je in principe op eendere welke server doen waarop je via SSH kunt inloggen. Wij zullen gebruik maken van GitHub. Github is een repository hosting waar je gratis open source projecten kunt delen. Projecten van gratis accounts zijn altijd publiek toegankelijk iedereen kan in principe aan je code (wel read-only).

Je kan andere github users ook schrijfrechten geven op jouw repository. Zo kunnen zijn ook wijzigingen aanbrengen en pushen naar jouw repository.

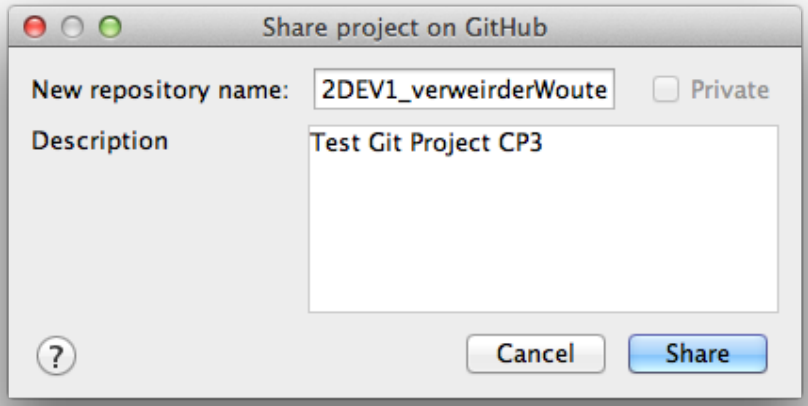
Je kunt ook abonnementen afsluiten om projecten private te zetten: dan kunnen enkel gebruikers die jij kiest toegang hebben tot het project.

We zullen als test eens ons project op github zetten. Maak een github account aan indien je dit nog niet gedaan hebt.

Zorg ervoor dat je de GitHub activeert in IntelliJ. Klik daarna in de menubalk op VCS → Import Into Version Control → Share project on GitHub.



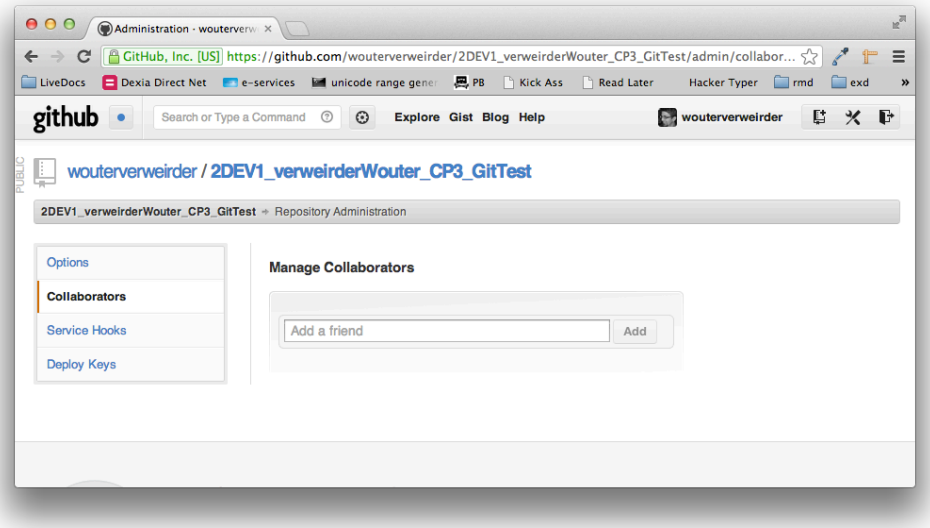
Vul je logingegevens in en klik op Login. Vervolgens kun je je project nog een description geven en sharen:



Neem een kijkje op github, je zou jouw project moeten zien staan. Nu kunnen ook andere mensen je project uitchecken.

SAMENWERKEN AAN ÉÉN REPOSITORY

Op de Github site van je project kun je andere mensen inviten om samen te werken op jouw repository. Klik op Admin → Collaborators:

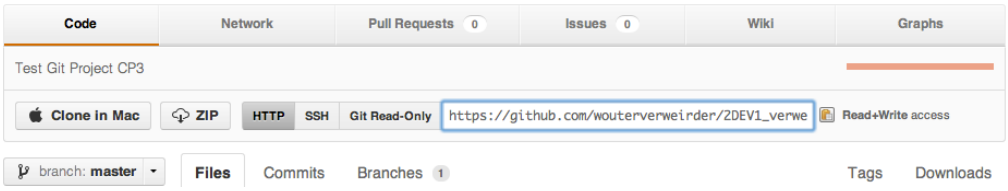


Voeg één van je klasgenoten toe als collaborator aan jouw project. Hij zal het project zien verschijnen in zijn lijst van repositories en kan starten met samenwerken.

EEN PROJECT UITCHECKEN

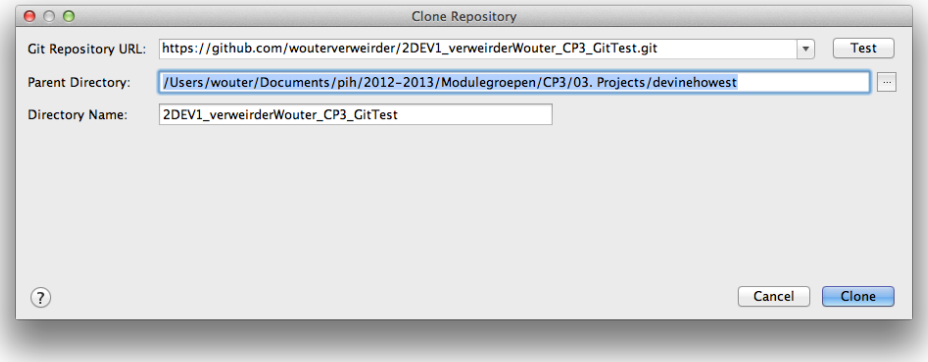
Iemand heeft jouw nu toegang gegeven tot zijn project op github. Nu moet je dit project nog binnentrekken op jouw computer, zodat je ook effectief kan meeprogrammeren.

Kopieer op de projectpagina van je collega de .git url. Deze vind je in het tekstveld en begint met https://github.com/accountnaamvancollega/...



Kies in IntelliJ voor VCS → Checkout From Version Control → Github

Als Git Repository URL geef je de .git url in:



De volledige repository (inclusief historiek, ...) wordt nu gecloned naar jouw computer. IntelliJ stelt voor om het project meteen te openen, je mag dit ook doen.

COMMITS DELEN

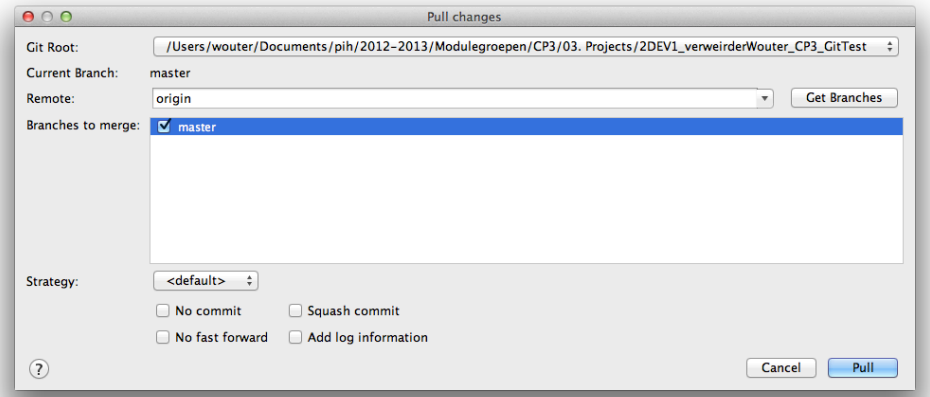
Je zal nu wat code aanpassen en committen naar de repository van je collega. Verwijder in het opstartbestand bijvoorbeeld de code die het tekstbestand aanmaakt, en commit dit. Dit zal de wijzigingen lokaal opslaan in de repository.

Je kunt zoveel wijzigingen lokaal committen als je wil. Zolang deze niet "gepushed" worden naar de repository op github, heeft maar 1 user deze staan. Wanneer je je wijzigingen wil delen met de andere users (publiceren op github) dan klik je rechts op je module → Git → Repository → Push. Hiermee zal je jouw wijzigingen aan de repository van je collega naar Github versturen.

UPDATES BINNENHALEN

Bij het pushen van jouw commits naar GitHub krijgen de andere collaborators niet automatisch de wijzigingen te zien op hun computer. Men moet deze wijzigingingen manueel “pullen”.

Wacht tot iemand anders wijzigingen gepushed heeft naar jouw github repository. Wanneer dit gebeurd is, kun je deze wijzigingen afhalen (pullen) door rechts te klikken op je module → Git → Repository → Pull. Kies in het window bij “Branches to merge” voor master, om ervoor te zorgen dat de nieuwere commits vanop github toegevoegd worden aan jouw master branch:



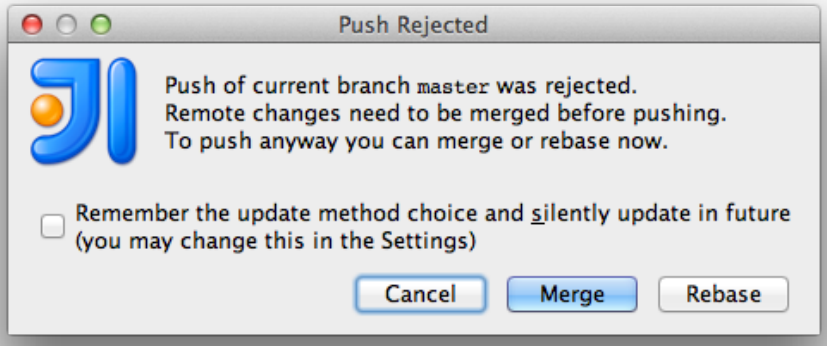
Mogelijk moet je na de pull actie nog rechtsklikken op je module en kiezen voor Synchronize… om de wijzigingen in je filesysteem in te laden.

CONFLICTEN OPLOSSEN

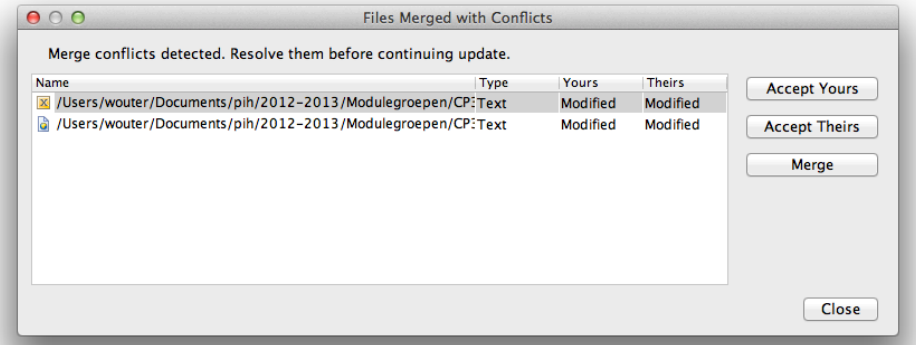
Wanneer je met meerdere mensen samenwerkt, is het onvermijdelijk dat je in dezelfde files tegelijk bezig bent en er conflicten optreden.

Laat een collega in je opstartklasse de stage alignement instellen en pushen naar jouw repository.

Pas dan in jouw files, zonder de changes te pullen je opstartklasse aan met code om de scalemode in te stellen, en probeer te committen & pushen naar github. Je krijgt de melding dat er nieuwere wijzigingen zijn op github en dat je deze eerst moet mergen met jouw files:



Kies voor Merge. Jullie hebben beide in dezelde file een andere lijn code geschreven. Git weet niet hoe hij dit automatisch moet samenvoegen en geeft je enkele keuzes om te mergen:

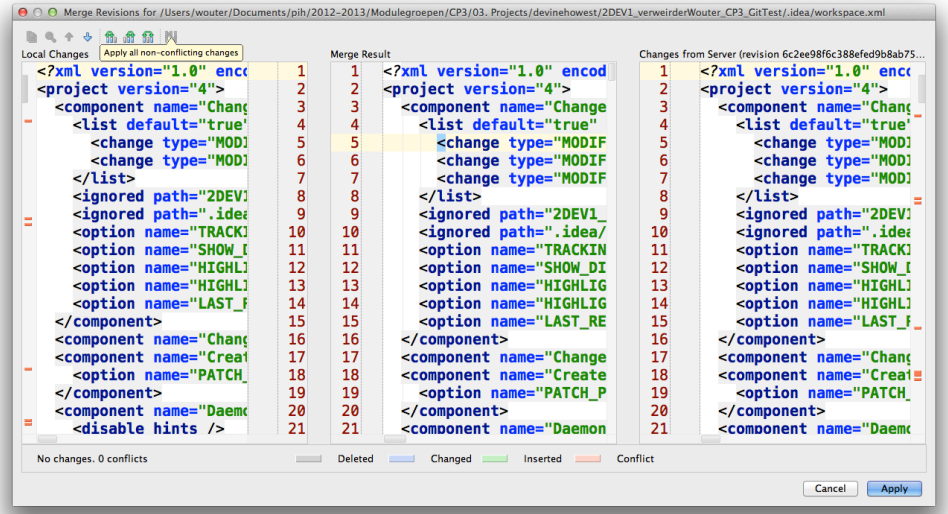


Accept Yours zal jouw versie als de juiste beschouwen, Accept Theirs zal de versie van de repository als de juiste beschouwen en Merge zal je toelaten om manueel aanpassingen te doen.

In ons geval hebben we 2 conflicten (kan verschillen met jouw situatie). Een conflict op één van de project XML files, en een conflict op ons opstartbestand.

We zullen kiezen voor Merge, aangezien dit de actie is waarbij je zelf nog manueel moet aanpassen. Je krijgt een scherm met 3 editors: jouw versie, een versie waarin je de aanpassingen zal doen en de versie vanop github.

In het geval van een merge op de XML file zijn heel wat verschillen die automatisch toegepast kunnen worden. Hiervoor klik je in de toolbalk op “Apply all non-conflicting changes”:



In het geval van de actionscript file, kun je zelf de nodige logische wijzigingen aanbrengen in de middenste editor en klikken op Apply.

Wanneer alle conflicten opgelost zijn kun je verderwerken.

GROEPSPROJECT OPZETTEN

Wanneer je alle bovenstaande zaken getest hebt, dan heb je de meeste git acties gebruikt die je nodig zal hebben, en kan je van start gaan met het groepswerk. Één van de teamleden zal een nieuw project aanmaken en dit op GitHub plaatsen. Spreek af wie dit doet. Deze persoon geeft het andere teamlid/teamleden ook toegang tot de repository, zodat iedereen het project lokaal kan binnenhalen en commits kan pushen naar GitHub.

Je hebt met je groep een naam toegewezen gekregen, bestaande uit het project (ibook of presentation engine) en een nr. Gebruik die naam als naam voor je project / repository. (bvb PRESENTATION01, IBOOK01, ...)